# Series 16

# HARDWARE DOCUMENTATION

**H316**
**Central Processor Description**

**INSTRUCTION MANUAL**

This manual contains a description of the H316 Central Processor and its operation. Information is also presented on data and command word formats, basic modes of operation, instruction fetching, and address modification. Refer to the H316 Central Processor Instructions and Logic Diagrams Manual, Doc. No. 70130072174, Order No. M-493, for detailed flow charts and analyses of commands.

DOC. NO. 70130072176C ■ ORDER NO. M-492 ■ APRIL 1973

Honeywell

REVISION HISTORY

| New Revision Level of Manual | Change No. | Effective Date | Number and New Revision Level of Affected Drawings | Pages Affected by Revision |
|---|---|---|---|---|
| B | 9689 | July 15, 1971 | --- | iv, 1-1, 1-5, 1-11, 1-12, 1-13 |
| C | 30014 | April 1973 | --- | iii, 1-1, 1-2, 1-3, 1-11, 1-12, 1-13, 2-4 thru 2-10 |

Publications Department, Field Engineering Division, Newton, MA 02161

Printed in the United States of America

CONTENTS

ILLUSTRATIONS

TABLES

# SECTION I
# INTRODUCTION

## SCOPE OF MANUAL

This manual contains the principles of operation and physical description for the H316 Central Processor Unit, Types 316-01, 316-0100, and 316-0110. These types differ primarily in mechanical design and configuration. Type 316-0110 contains the improved memory module.

## APPLICABLE DOCUMENTS

All manuals provided as standard documentation with each H316 computer are listed in Table 1-1.

The documentation package provided with each H316 system includes all the documents listed in Table 1-1. Customers may obtain additional copies of any manual by contacting their local Honeywell representative or by writing directly to:

Honeywell Information Systems Inc.
Framingham Computer Operations
Old Connecticut Path
Framingham, Massachusetts 01701

Table 1-1.
Standard H316 Documentation

| Title | Document No. |
|---|---|
| Honeywell 316/516 Programmers Reference Manual | 70130072156 |
| Honeywell 316/516 Operators Guide | 70130072165 |
| H316 Central Processor Description | 70130072176 |
| H316 Central Processor Instructions and Logic Diagrams | 70130072174 |
| H316 Circuit Modules and Parts | 70130072166 |
| H316 Interface Manual | 70130072167 |
| H316 Central Processor Installation Manual | 70130072205 |

## GENERAL DESCRIPTION

The H316 computer is a solid-state, 16-bit binary word, general purpose computer with an internally stored program, a 1.6-$\mu$s memory cycle time, and a memory expandable from 4 to 32K. The machine has a fully parallel organization and both indexing and multi-level indirect addressing capabilities. Standard features include a flexible repertoire of 72 commands, a powerful input/output (I/O) bus structure, standard teletype I/O equipment, and a full line of options and optional peripheral devices.

The 16-bit word allows a straightforward and efficient "sectorized" addressing scheme. The use of large sectors permits most instructions to be coded in one word each. The 16-bit machine word is directly compatible with the ASCII 8-bit character code.

The overall characteristics of the H316 computer are given in Table 1-2.

Table 1-2.
Leading Particulars

| | |
|---|---|
| Primary Power | 475 watts, 5.5 amperes at 115 vac ±10% at 60 ±2 Hz |
| Type | Parallel binary, solid state |
| Addressing | Single address with indexing and indirect addressing |
| Word Length | 16 bits (single precision)<br>31 bits (double precision) |
| Machine Code | Two's complement |
| Circuitry | Integrated |
| Signal Levels | Active: 0v<br>Passive: +6v |
| Memory Type | Coincident-current ferrite core |
| Memory Size | 4K to 32K in 4K or 8K modules |
| Memory Cycle Time | 1.6 μs |
| Instruction Complement | 72 instructions |
| Speed | |
| Add | 3.2 μs |
| Subtract | 3.2 μs |
| Multiply (optional) | 8.8 μs |
| Divide (optional) | 17.6 μs |
| Standard Memory Protect | Designed to protect memory data in the event primary power fails |
| Standard Interrupt | Single standard interrupt line |
| Input/Output Modes | Single word transfer<br>Single word transfer with priority interrupt<br>Data multiplex channel (optional) |
| Standard I/O Lines | 10-bit address bus - (4 function code and 6 device address)<br>16-bit input bus<br>16-bit output bus, priority interrupt external control and sense lines |
| Standard Teletype | Read paper tape at 10 cps<br>Punch paper tape at 10 cps<br>Print at 10 cps<br>Keyboard input<br>Off-line paper tape preparation, reproduction, and listing |
| Dimensions: | |
| Standard rack mountable | (See Figure 1-1) |
| Table top | 15.47 high, 17.69 wide, 26 in. deep (see Figure 1-2) |
| Pedestal model | (See Figure 1-3) |
| Rack mounted | Same as standard but contained in a 24 x 24 x 72 in. rack. |
| Weight: | |
| Standard rack mountable | 120 lb |
| Table top | 150 lb |
| Environment | Room ambient for computer less I/O devices: 0° to 45°C |
| Cooling | Filtered forced air |

Computer Operations

The main access paths for data and command words in the H316 computer are shown in simplified form in Figure 1-4. As shown, the computer consists of control logic for the development of clock, control, and enable levels; memory for the storage of data, and a series of registers (M, X, P, etc) which are used for actual handling and processing of data. The computer is also supplied with a control panel which is used for the manual entry of data into the computer, the control of the operations to be performed, and the display of data and operational information.

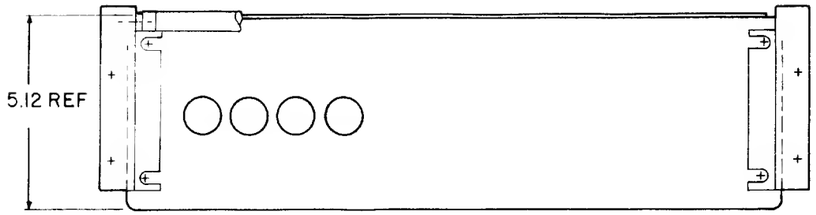
The gating of data between the registers of the computer is performed by adder gating logic and by the D register. Data transfers to and from the computer are performed by an I/O structure consisting of:

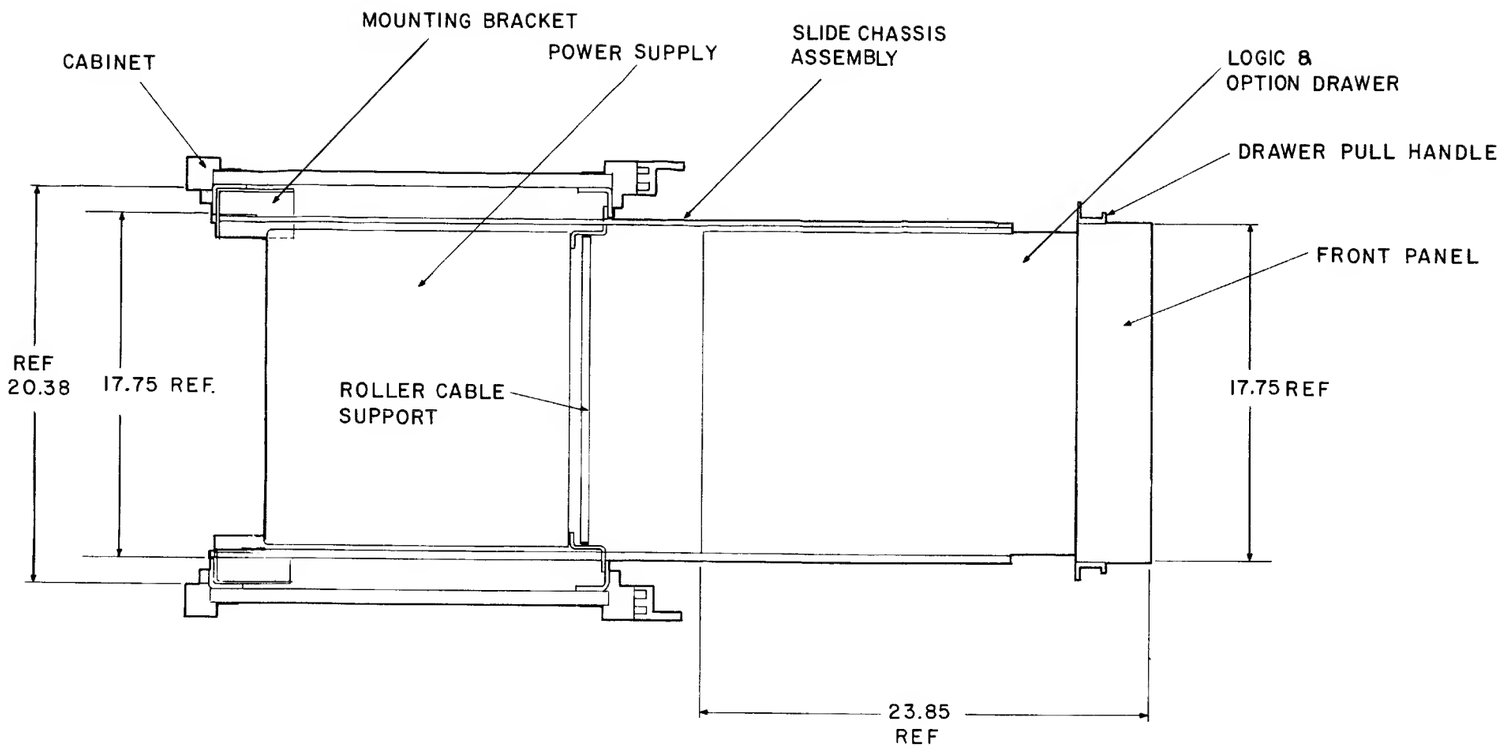
an address bus (ADB)
an output bus (OTB)
an input bus (INB)
control signals.

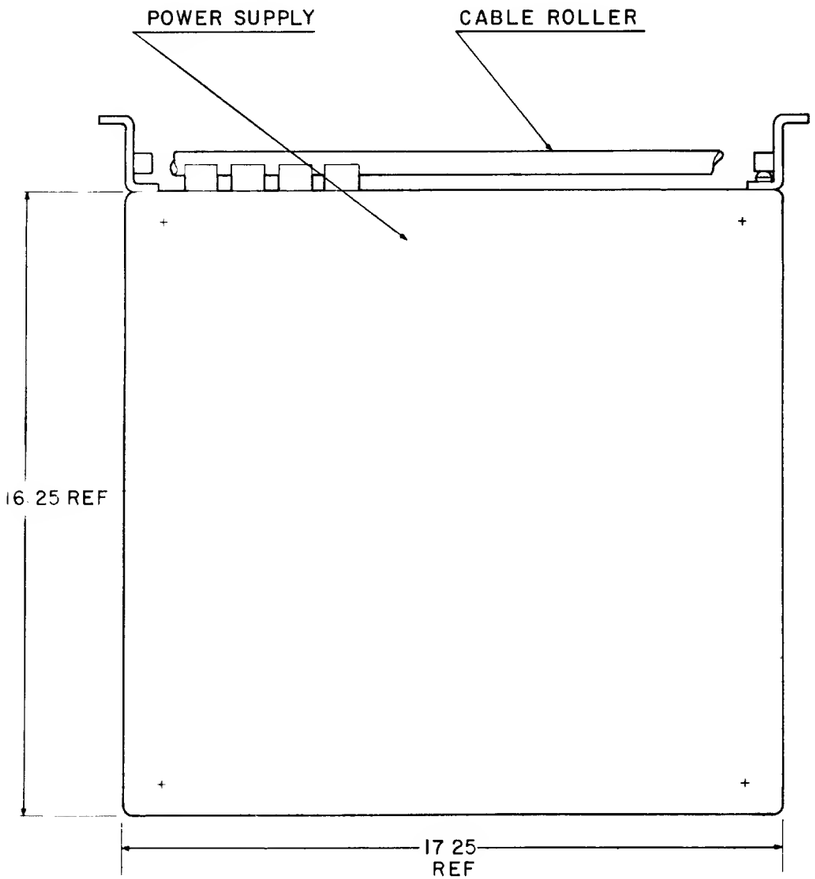
## PHYSICAL DESCRIPTION

The standard H316 computer (see Figure 1-5) is a rack mountable unit and consists of a main frame assembly, fixed power supply assembly, control panel, and the necessary hardware for installation in a standard 19 x 24 inch rack. In addition, the H316 computer can be obtained in a rack mounted unit, which is identical with the standard unit but with a Honeywell supplied rack, a table top unit, and a pedestal model (Figures 1-6, 1-7, 1-8 respectively). See Figures 1-1 through 1-3 for the dimensions of each of these configurations.

Refer to Installation Manual Document No. 70130072205A for installation and checkout procedures.



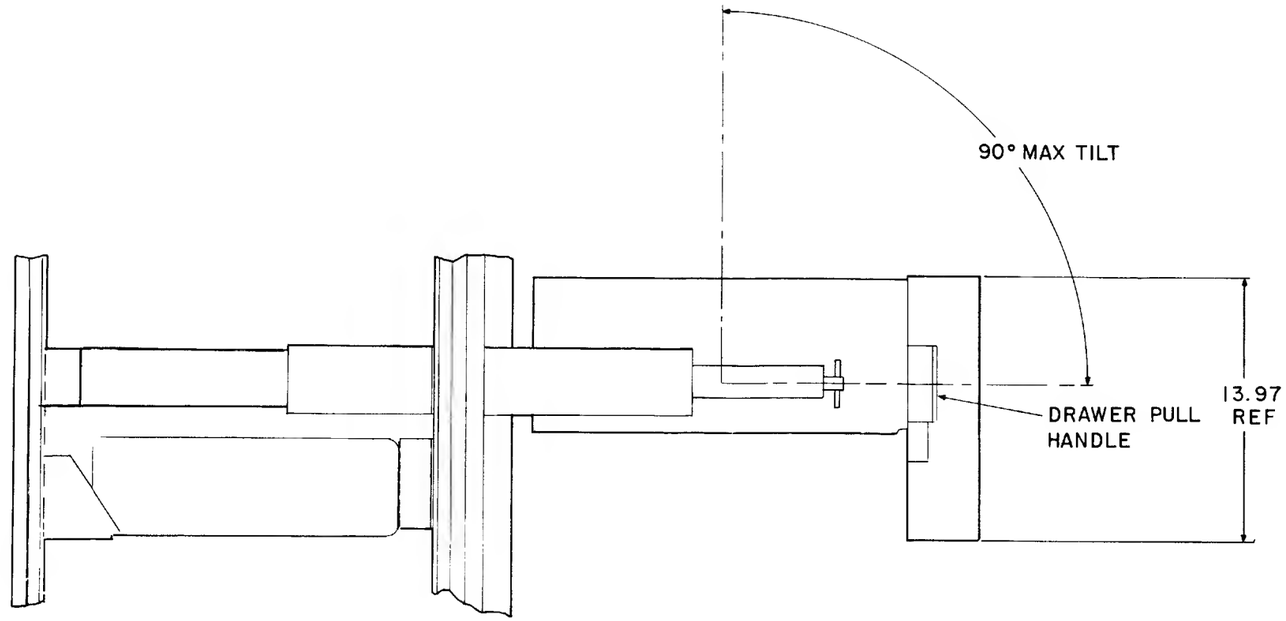




Figure 1-1. H316 Rack Mountable Unit, Dimensional View

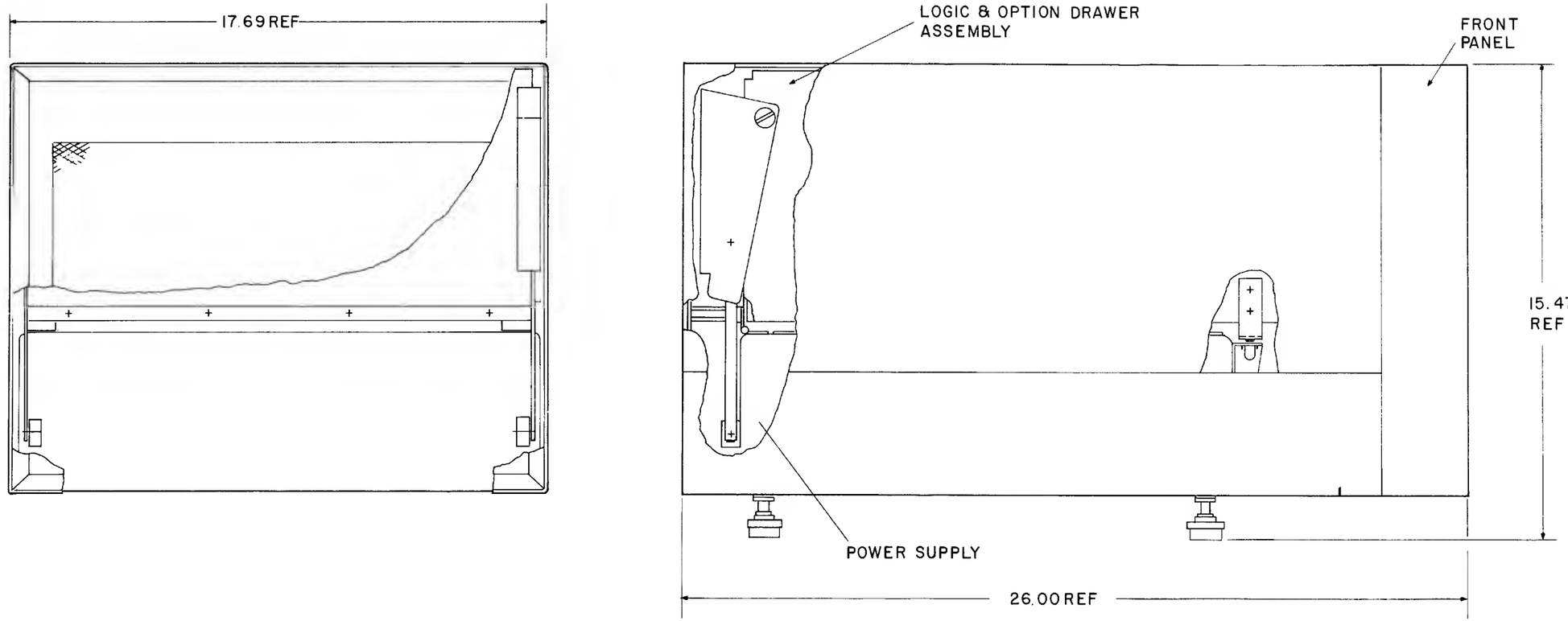



Figure 1-2. H316 Table Top Unit, Dimensional View

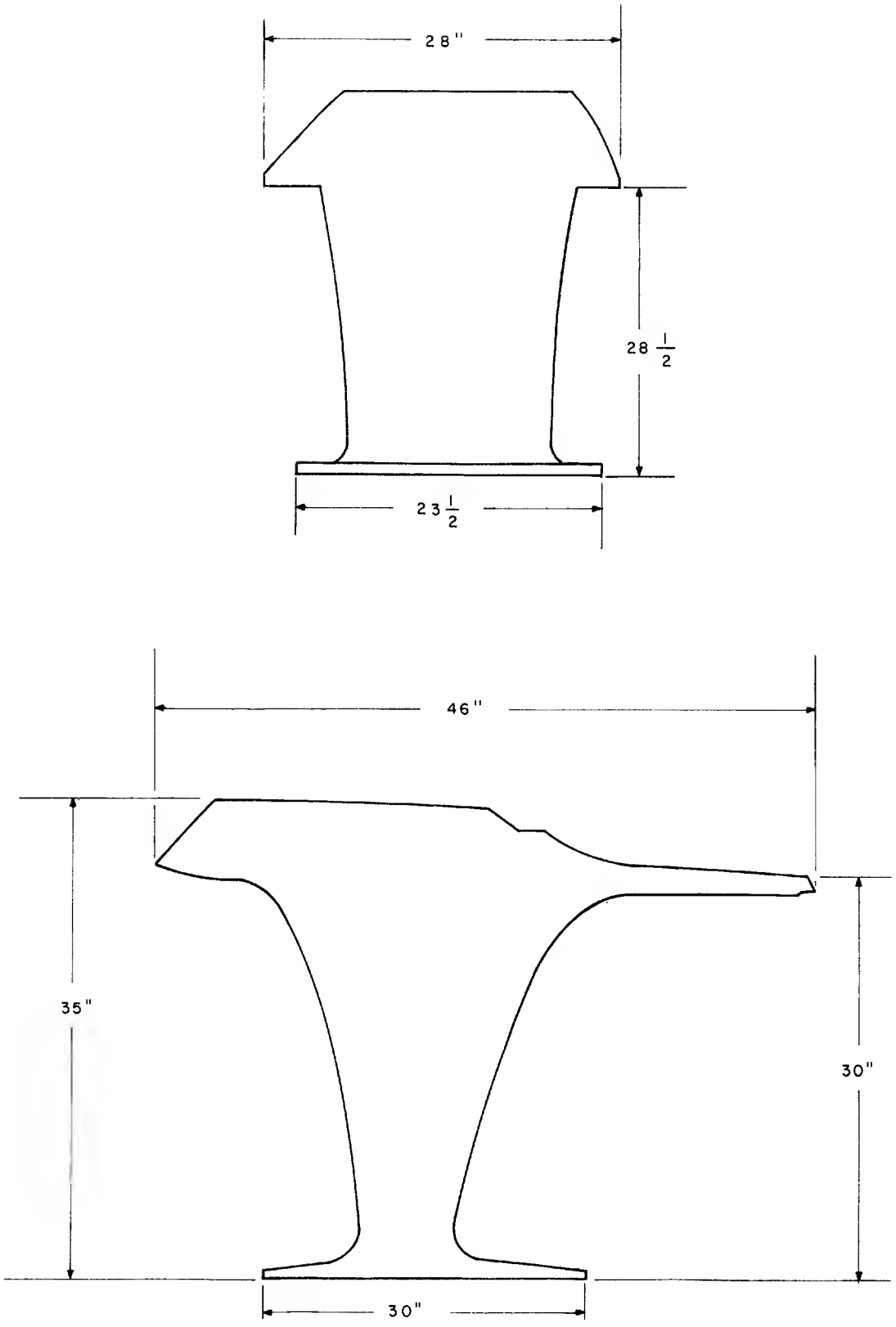



Figure 1-3. H316 Pedestal Model, Dimensional View

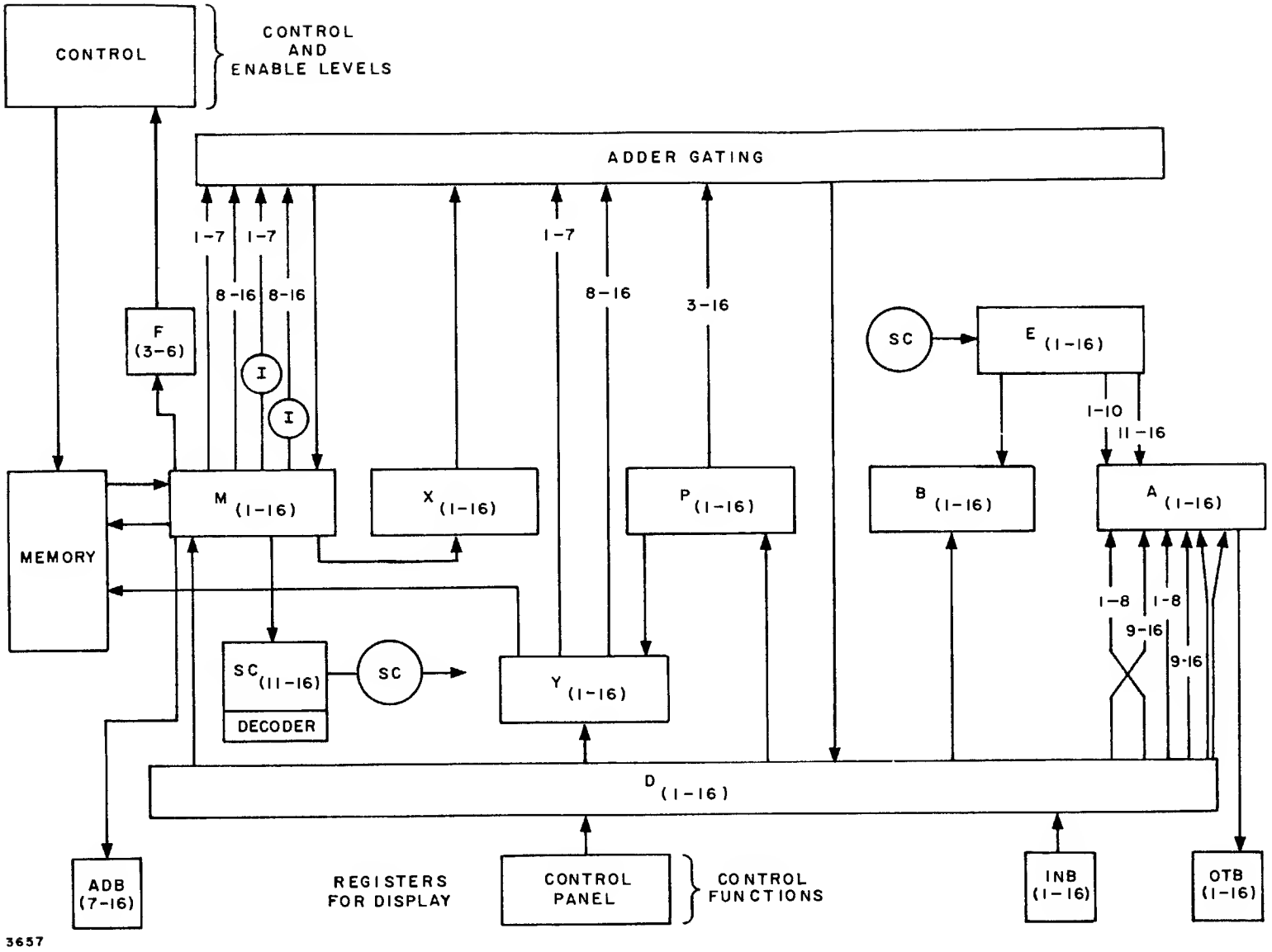



Figure 1-4. H316 General Purpose Computer, Simplified Block Diagram

71-00358

Type 316-01

Types 316-0100 and 316-0110

Figure 1-5. H316 Rack Mountable Configuration

71-0035A

Type 316-01

Types 316-0100 and 316-0110

Figure 1-6. H316 Rack-Mounted Configuration

Type 316-01

5765


71-0020

Types 316-0100 and 316-0110

Figure 1-7. H316 Table Top Configuration

5954

Figure 1-8. H316 Pedestal Unit Configuration

# SECTION II
# THEORY OF OPERATION

The following information is organized to supplement and complement the flow charts, instruction analyses, and the logic diagrams which are located in the H316 Central Processor Instructions and Diagrams manual. A discussion of the overall operational sequences is based on a master flow chart that is a condensation of the fully detailed flow charts which are located in the H316 Central Processor Instructions and Diagrams manual. Included in this section are discussions dealing with data and command word formats, basic modes of operation, and the processes involved in instruction fetching, address modification, and execution.

## CENTRAL PROCESSING UNIT (CPU)

A discussion of the CPU data flow, based on an overall block diagram, introduces the control signals that appear most frequently on the flow charts of the H316 Central Processor Instructions and Diagrams manual. Complex logic structures, such as the adder and clock system, are described in detail.

## FORMAT AND EXECUTION OF INSTRUCTIONS

### Word Structure

Data Words. -- Data words are stored in binary form using two's complement notation. The H316 accepts and processes data words in both single and double precision. Single-precision data words (Figure 2-1) include 15 magnitude bits plus a sign bit and represents a data range of $\pm 2^{15}$ or $\pm 32,768$.

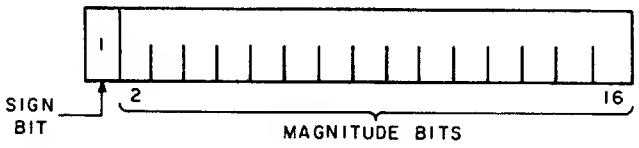



Figure 2-1. Data Word Format, Single Precision

A double precision number (Figure 2-2) consists of two data words, each one having 15 magnitude bits. The first data word includes the 15 most significant bits (MSB) of the double precision number plus a sign bit. It is identical to a data word using single precision. The second data word includes the 15 least significant bits (LSB) of the double precision word. The sign position is always zero. Double precision data words represent a data range of $\pm 2^{30}$ or ±1, 073, 741, 824.

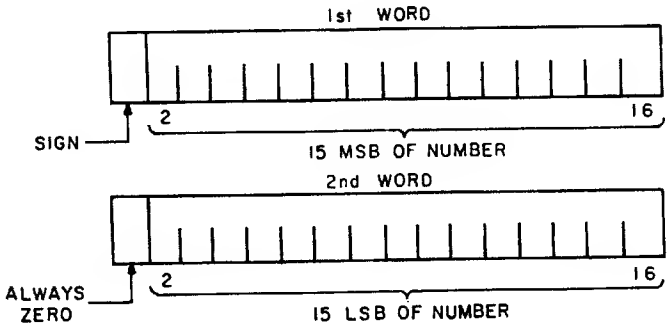



Figure 2-2. Data Word Format, Double Precision

Instruction Words. -- There are four types of instruction words:

Memory reference
Generic
Input/Output
Shift

The memory reference instructions are identified by the format shown in Figure 2-3. Bit 1 is the flag bit used to specify indirect addressing; bit 2 is the tag bit used to specify indexing; bits 3 through 6 are the operation code (op code) field; bit 7 is the sector bit; and bits 8 through 16 are the address field.

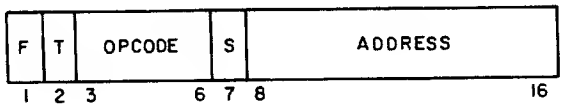



Figure 2-3. Memory Reference Instruction Format

Generic instructions are identified by the format shown in Figure 2-4. Bits 1 through 16 are used as the op code.

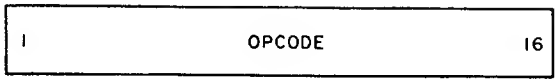



Figure 2-4. Generic Instruction Format

Input/output instructions are identified by the format shown in Figure 2-5. Bits 1 through 6 contain the op code and bits 7 through 10 specify the type of function to be performed. The I/O device is specified by bits 11 through 16.

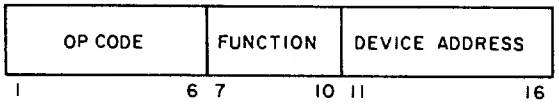
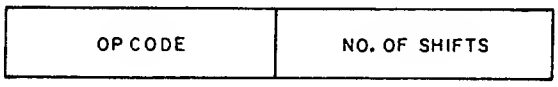



Figure 2-5. Input/Output Instruction Format

The shift instructions are identified by the format shown in Figure 2-6. Bits 1 through 10 contain the op code and bits 11 through 16 contain the two's complement of the number of shifts to be performed.




3650

Figure 2-6. Shift Instruction Format

## MEMORY SECTORS AND SPECIAL LOCATIONS

Figure 2-7 is an illustration of the memory sectors in a 4096 word memory. Each sector has 512 words. There are 15 dedicated memory locations reserved for a fill program. They are located at $(00001)_8$ through $(00017)_8$. The contents of these locations can only be altered by using the memory access mode from the control panel. The logic which protects these locations is shown on LBD 126.

A power failure causes an interrupt to location $(00060)_8$ and the standard interrupt link is at location $(00063)_8$ (the logic is shown on LBD 135).

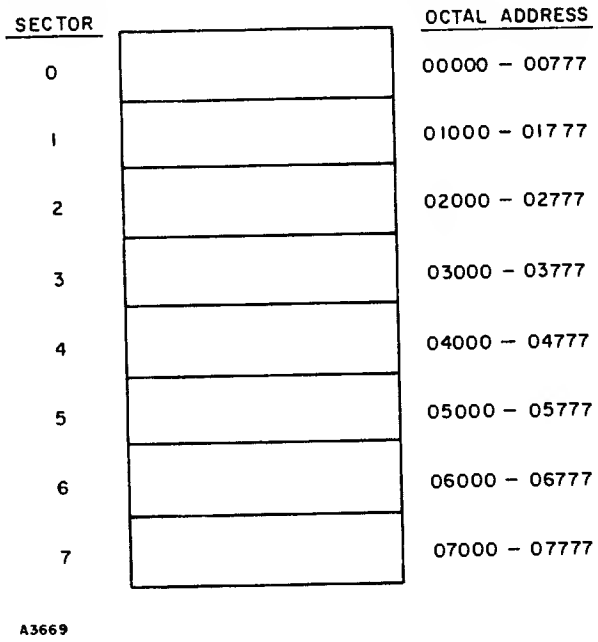



Figure 2-7. Memory Section in a 4096-Word Memory

## EFFECTIVE ADDRESS FORMATION (Figure 2-8)

The standard memory reference instruction format uses 10 bits (7-16) for memory addressing purposes. Nine of the bits (8 through 16) are used to address any location within a specified sector. Bit 7 specifies the sector. When bit 7 is a 1, the specified sector is the current sector in which the instruction is located; when it is a 0, the specified sector is sector zero or the base sector. Two addressing modes are used for address formation: the normal addressing mode and the extended addressing mode.

### Normal Addressing Mode

The normal addressing mode is used exclusively with systems that have memory sizes of 16, 384 words or less where no memory expansion is contemplated. Any memory address in this range may be specified by the normal 14 address bits (3 through 16). Bits 8 through 16 specify the address within a sector, and bits 3 through 7 specify the sector. Bits 1 and 2 will be 0.

After determination has been made that the machine is not operating in the extended addressing mode (i.e., is operating in the normal addressing mode), first the sector bit (7)

and then the index bit (2) are examined. If sector zero is indicated, bits 1 through 7 of the Y register are 0's; if the current sector is indicated, bits 1 through 7 are left unchanged. If indexing is indicated, the contents of the index register are added into the final value of the address.

The indirect address bit (1) is next examined. If indirect addressing is indicated, the contents of the specified address are treated as an address, and a new round of checking for indexing and indirect addressing takes place. Any number of indirect addressing and indexing levels are permitted unless memory lockout is installed. In that case, only eight levels are allowed.

When no indirect addressing is indicated or all indirect addressing operations have been performed, the contents of the specified address are treated as an instruction, and the instruction is executed.

Extended Addressing Mode (LBD 0.136)

The extended addressing mode is used with systems that have memory sizes ranging from 16,384 words up to and including 32,768 words. Systems that have 16,384 words of memory or less but may later require a memory expansion to greater than 16,384 words should include the option that allows operation in the extended addressing mode regardless of size. Hardware is added to increase the size of the P-register and Y-register by one bit (P02 and Y02) and to change the interpretation of the index bit of the indirect address word to provide a fifteenth address bit. Any memory address up to 32,784 may be specified by 15 address bits (2 through 16). Bits 8 through 16 specify the address within a sector, and bits 2 through 7 specify the sector. Bit 1 will be 0.

The control logic for extended addressing is shown in LBD 0.136 of the Instructions and Logic Diagrams Manual, Doc. No. 70130072174. An extended mode flip-flop (EXTMD) determines whether the machine is operating in the extended mode or not. Entry to and exit from the extended mode is under program control, but entry is also automatic through the occurrence of a program interrupt. A monitor flip-flop (PMIND) retains the mode in which the computer is operating if an interrupt occurs.

A program may operate either completely in the upper half of memory (greater than 16,384 words) or completely in the lower half of memory (less than or equal to 16,384 words) with the extended mode disabled but will not be able to cross the upper/lower half boundary until the extended mode is enabled.

Only one level of indexing is possible in the extended mode. It is specified by bit 2 of the instruction word and is always the final operation in generating the effective operand address. It is forced to occur after indirect addressing as a function of signal M02DJ+. Signal M02DJ+ replaces M02FF+ at the input of EXSTL- (LBD 0.128) of the central processor. EXSTL is used to enable the contents of the X-register to the summand G in the central processor sum network, and this operation is inhibited when extended mode is enabled during the following specific conditions:

a. When the indirect LDX instruction is executed.
b. When indexing was not called for in the original instruction word.
c. When indirect addressing is called for.

When the extended mode is not enabled, the indexing and indirect addressing operations function normally.

Signal BSICY- prevents M02 from controlling the adder during an I-cycle while the extended mode is disabled. Similarly, signal EXTMD- prevents M02 from entering the adder during the A-cycle of a JST instruction while the extended mode is enabled. Signals H02DJ+ and H02DJ+A are additional inputs from P02 to summand H of the central processor sum network. BSH02- is added to EP02D- (LBD 0.102) to force the replacement of M02 by P02 when the extended mode is enabled.

Signal P02BS+ controls the state of Y02FF via signal BSD02+, which replaces D02FF+ in extended mode operation. P02BS is updated every fetch cycle regardless of whether or not the extended mode is enabled or disabled. It thus reflects the condition of Y02FF during the previous fetch. Only with the extended mode enabled can Y02FF differ from P02BS, thereby allowing the upper/lower half memory boundary to be crossed.

When an interrupt occurs, the forced JST instruction that is executed generates signal CLRF5-, which sets the state of the extended mode flip-flop (EXTMD) and allows the state of the extended mode control flip-flop (SEXTF) to be transferred to the previous mode indicator (PMIND). The state of PMIND is then stored in the A-register of the central processor as bit 03 during an input keys (INK) instruction. During the subsequent return-from-interrupt sequence, the state of SEXTF is restored when an output keys (OTK) instruction is executed. This state is transferred to EXTMD during TL1 time.

When it has been determined that the processor is operating in the extended mode, the order of performing the indexing and indirect addressing operation is reversed. The indirect address bit (1) and then the sector bit (7) are examined. If sector zero is indicated, bits 1 through 7 of the Y-register are 0's; if the current sector is indicated, bits 1 through 7 are left unchanged.

If indirect addressing is not indicated, then the index bit (2) is examined. If indexing is indicated, the contents of the index register are added into the final value of the address. If, however, indirect addressing is indicated, no indexing is allowed until all levels of indirect addressing have been completed. Then the index bit (2) of the original instruction is examined, and if indexing was indicated, the contents of the index register are added to the final value of the address. Any number of indirect addressing levels are permitted unless memory lockout is installed. In that case, only eight levels are allowed. Only the one level of post-indexing is permitted in any case, regardless of the number of indirect address levels.

When all operations have been performed, the contents of the specified address are treated as an instruction, and the instruction is executed.

## Memory Parity (LBD 0.133)

The memory parity option establishes an odd parity check function in the computer memory system. The hardware enables the generation of an odd parity bit for each memory input word, provides storage in memory for the added parity bits, checks the parity of each word read from memory, and gives an error indication when an error has been detected.

During a clear-write cycle, the number of 1's in the word being written into memory is determined. If the number is odd, a 0 will be written into the parity bit. If the number is even, a 1 will be written into the parity bit instead.

During a read-regenerate cycle, the parity of the word being read out of memory is determined. This determined parity (PAGED+) is then compared with the stored parity (PAMEO+) in the parity bit for that word. If they compare, the word is error-free. If they do not compare, then an error has occurred in one or more bits of the word. An error flip-flop (MPEFF) is then set by the parity error strobe (EPARB+), which is present only during the read-regenerate cycle (RRCXX+). Signal MEMAC- prevents parity checking during a fetch operation from the panel.

If the parity mask flip-flop (MPAFF) is set, the setting of MPEFF will generate an interrupt request on the priority interrupt line (PIL00). The resultant interrupt will be handled in the normal way. The programmer-generated interrupt service routine will determine what is to be done when an error has occurred. MPAFF will be set if bit 15 of the A-register is a ONE during the execution of a SMK0020 instruction and reset if bit 15 is a ZERO.

The state of MPEFF may also be interrogated under program control via the skip-if-set (SPS) and skip-if-reset (SPN) instructions (see Programmers Reference Manual for details). These instructions do not change the state of MPEFF. MPEFF can be reset only by the RMP instruction or by a master-clear operation. A light in bit position 15 on the control panel is also lit when MPEFF is set and the OP-register is selected.

## INSTRUCTION PHASES, PHASE SEQUENCING, AND OPERATING MODES

Program instruction processing (refer to Figure 2-9) requires from one to three types of machine phases. These phases are called F, A, and I. Instructions are composed of an integral number of phases, where each phase sets up the following one, depending on the instruction being performed.

Every instruction has an F phase. This phase fetches the instruction to be performed and performs indexing if the instruction word calls for indexing. If the instruction word calls for indirect addressing, the F phase sets up an I phase.

The I phase uses the address generated by the F phase to fetch a new address and performs indexing if the new address calls for indexing. If the new address calls another indirect address, an additional I phase is set up.

All memory reference instructions except JMP have at least one A phase. It is during an A phase that operands are fetched or stored and/or operated upon. When multiple or extended A phases are necessary, the shift counter is used.

The operand address used by the A phase is called the "effective operand address" (EA). It is established in the previous F or I phase. The last A phase sets up the F phase for the next instruction.

All instructions are composed of one of the nine phase sequences shown below:

F → F

F → I → F

F → I → I. . . → I → F

F → A → F

F → I → A → F

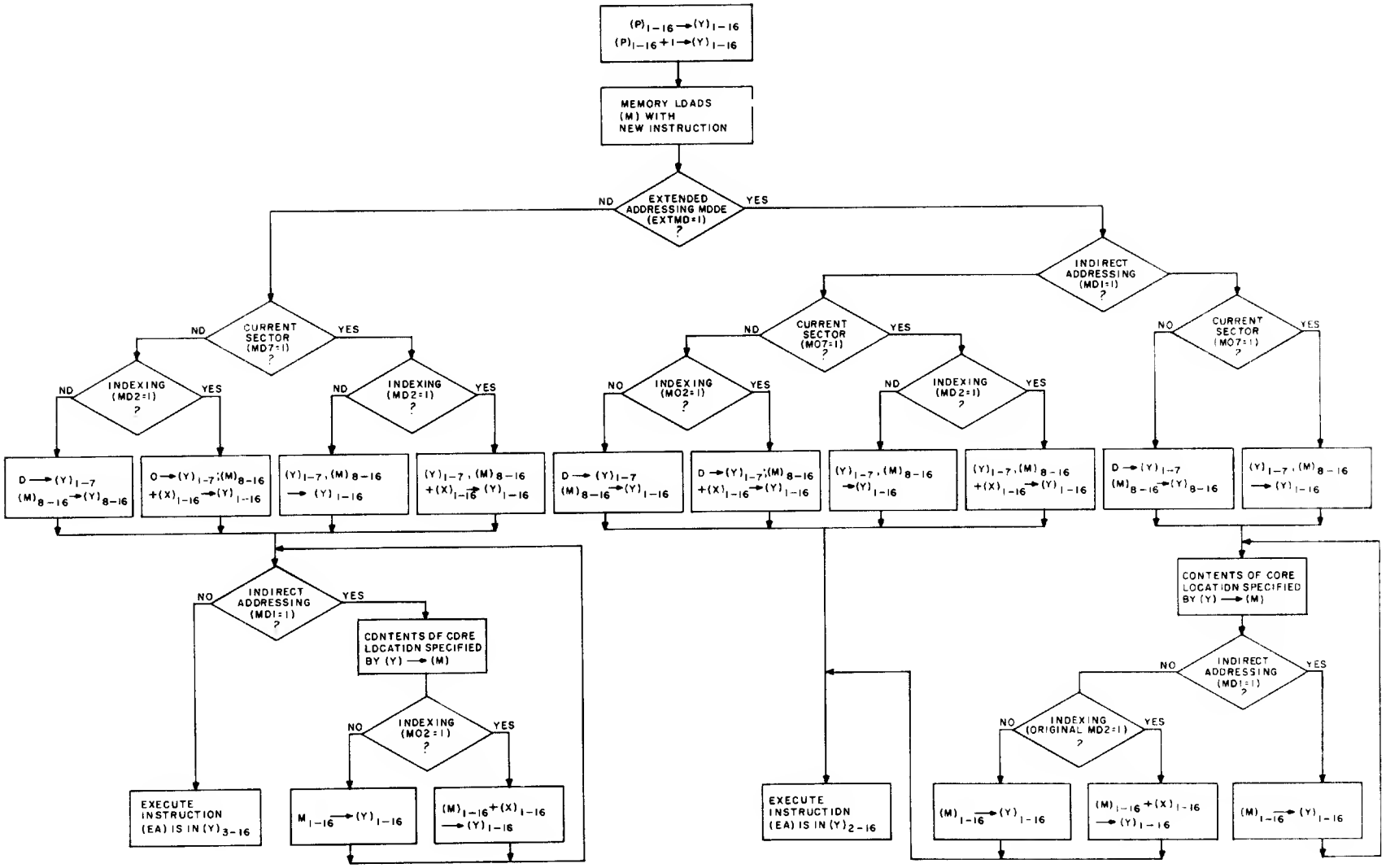



Figure 2-8. Effective Address Formation

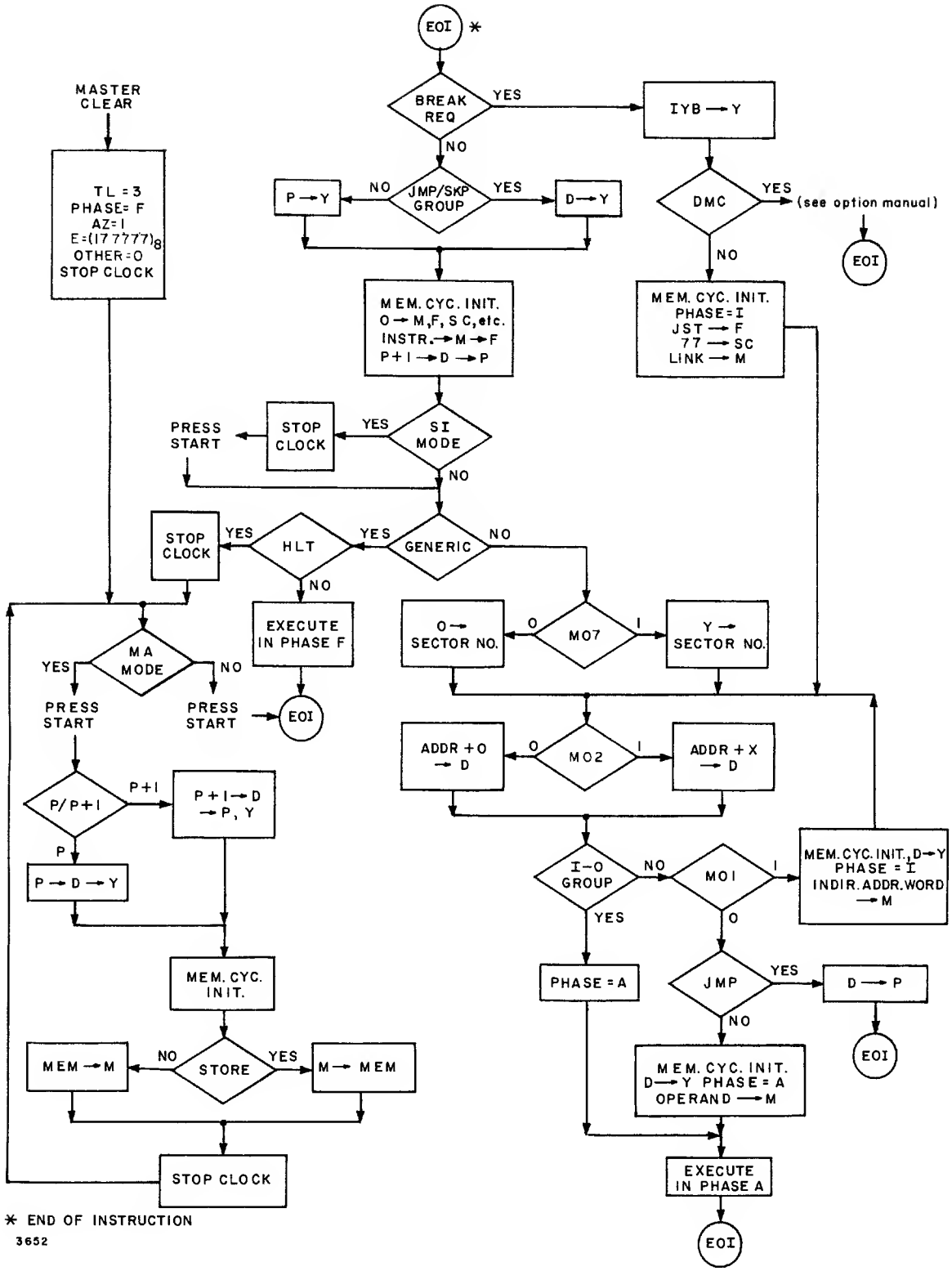



Figure 2-9. Basic Control Flow Chart

F → I → I . . . → I → A → F
F → A → A . . . → A → F
F → I → A → A . . . → A → F
F → I → I . . . → I → A → A . . . → A → F

The use of optional I/O devices can cause "breaks" and "interrupts" in the normal execution of a program in progress. A break is defined as an operation which interjects a function without altering the P register. An interrupt is defined as an operation which interrupts the normal sequencing of instructions being performed by altering the P register.

The computer breaks can be initiated by RTC (real-time clock), MI (memory increment), and DMC options. RTC and MI breaks can only occur when the CPU has completed an instruction. SI (standard interrupt) and PI (priority interrupt) can interrupt only when the CPU is in the "permit interrupt" status. PFI (power fail interrupt) can interrupt regardless of the "permit interrupt" status.

If the CPU is placed in the single instruction mode, sequential instructions can be executed one at a time each time the START button is depressed. Instruction execution can then be examined by using the front panel controls and indicators. Depressing the START pushbutton executes the most recently fetched instruction and fetches the next instruction.

When the operator desires to read and/or alter the content of any memory location, a memory access mode is initiated. Front panel controls and indicators permit the operator to display and alter memory locations. Consecutive locations can also be displayed and/or altered with the proper selection of front panel controls.

## CENTRAL PROCESSOR DATA FLOW

Figure 2-10 is a simplified diagram illustrating the flow of data to and through the central processor.

Note that the D register is the central register through which most data flow occurs, hence its designation as the D (distribution) register. Entry into memory, buffered by the M register, is possible either through the sum network to the D register or from the sum network directly to the M register. The reader should become familiar with the mnemonics at the inputs to the registers. All signals with an E for the first letter are enable signals to route data from one functional area to another. For instance, EAS (an abbreviated form of EASTL) means "enable the A register to the sum network," and EEA (an abbreviated form of EEALS/EEATS) means "enable the E register to the A register."

Other signals seen on Figure 2-10 are the SR/SL and the ENS signals. SRA, for instance, means "shift right to A register." ENS means "enable the negation of the M register to the sum network."

With regard to the sum network, two other points are worth mentioning. Note the input to the sum network with E1S16 and Y01 as inputs. This is used as a Compare A with Storage (CAS) instruction to skip one or two instructions based on the state of the Y01 bit. The other point is the function of signals EIK17 and JAMKN as they pertain to the sum

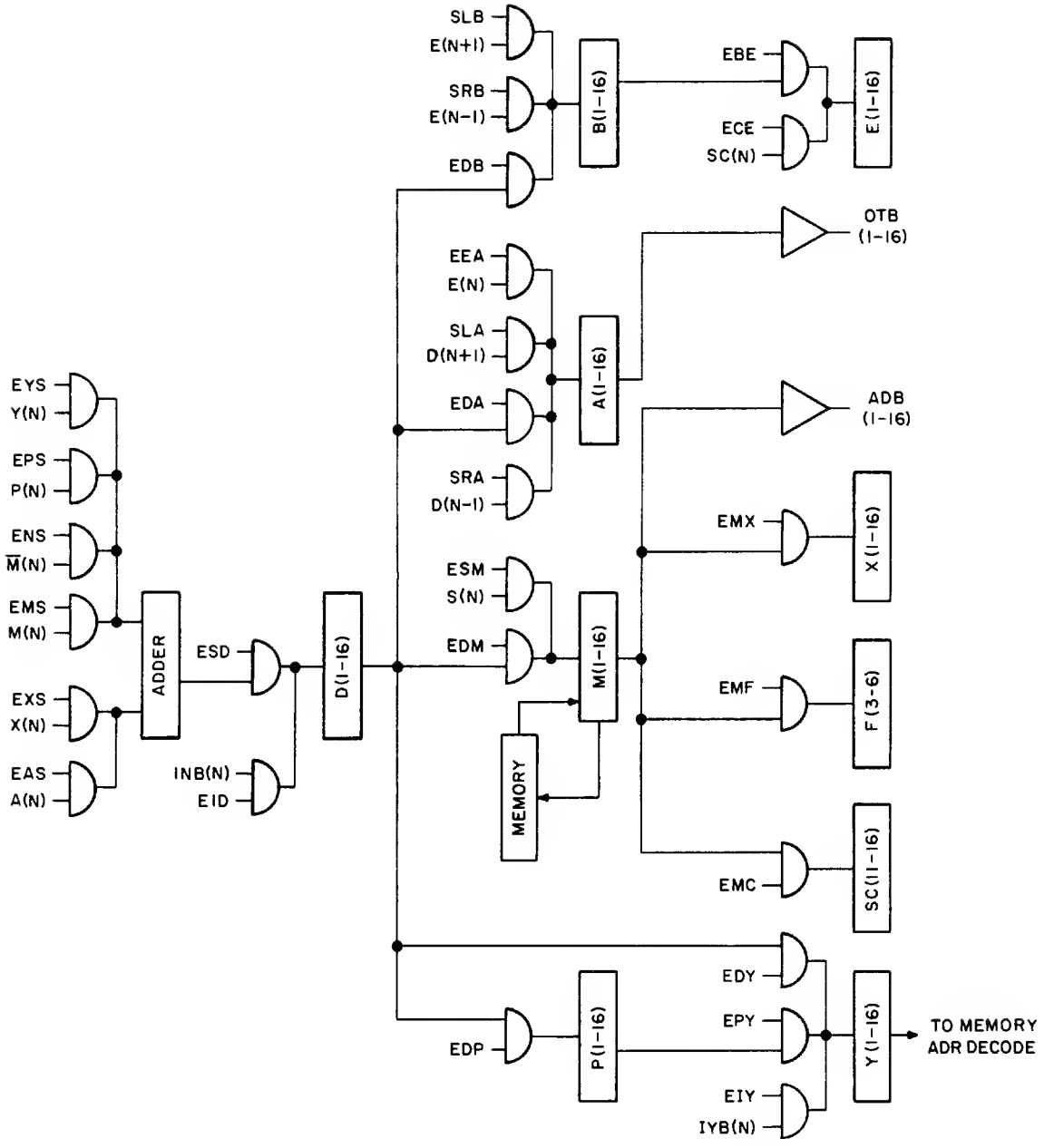



Figure 2-10. Control Processor Data Flow Chart

network. EIK17 is used to force a low-order carry and JAMKN is used to suppress all carries in the sum network.

The input bus has access to the central processor through the D register. Control signal EID (enable input bus to D register) gates the input bus information into the D register. The output bus requires no such control signals to gain access to the contents of the A register since it is always connected to the A register.

## FUNCTIONAL AREA DESCRIPTIONS

The following paragraphs contain descriptions of the column registers, the sum network (with examples of addition and subtraction), the clock system, the P register, the shift counter, data transfers, and operation decoding.

### Column Registers

A Register. -- The A register is a 16-bit register used as the primary arithmetic and logic register of the computer that can be displayed and manually controlled from the computer control panel.

B Register. -- The B register is a 16-bit secondary arithmetic register used to hold arithmetic operands which exceed one word in length that can be displayed and manually controlled from the control panel.

P Register. -- The P register contains the core address of the next instruction to be performed. Its contents are incremented by one each time an instruction is fetched and may be incremented an additional number of times during the execution of certain instructions. In the case of a jump instruction, the P register is loaded with the core address to which the program is to jump. During a Compare A with Storage instruction, the P register can be caused to skip one or two instructions. The P register can be manually controlled from the control panel.

Y Register. -- The Y register is a 16-bit memory address register that can be displayed and manually controlled at the control panel.

E Register. -- The E register is a 16-bit register used when shifting the B register.

D Register. -- The D register is a 16-bit register through which all local data flow in the central processor occurs. Transfers from the sum network to all other registers except for the M register which can be loaded directly from the sum network or the D register, are made through the D register.

F Register. -- The F register is a 4-bit register which retains the op code contained in bits M03 through M06 of the M register.

X Register. -- The X register is a 16-bit index register. Any memory cycle which alters the content of location zero also changes the X register.

Sum Network

For purposes of this discussion, the sum network is considered as consisting of four parts: the summand selection, the intermediate functions, the carry network, and the sum formation/ strobing networks. (See Figure 2-11.)

Summand Selection. -- With reference to Figure 2-12 and LBDs 101-116, note that there are two summands, G and H (GXXDJ and HXXDJ). Each of the summands is selected from among the several column registers (A or X for summand G and M, P, or Y for summand H), depending on the algorithm of the current instruction. (See Figure 2-12.) Summand G can be gated from the A register with enable level EASTL+ or from the X register with enable level EXSTL+. If no selection is specified, the quiescent state of summand G is zero. Similarly, summand H can be gated from the M register with enable levels EMSHL+ and EMSLL+, from the one's complement of the M register ($\overline{M}$) with ENSHL+ and ENSLL+, from the P register with EPSLL+, or from the Y register with EYSHL+ and EYSLL+.

Note that the enable levels for selecting the M and Y registers are divided into a high and low order part to facilitate split word operations. The high order part of the input includes bits 1 through 7 and the low order part includes bits 8 through 16.

Since summand H is complemented relative to summand G (note polarities of inputs from registers on LBDs 101 through 116), the absence of any input to summand H renders it $177777_8$ rather than zero. In some algorithms, more than one register can be simultaneously selected for the same bits of summand H. When this occurs, summand H becomes the logical product (AND) of the selected registers. If M and $\overline{M}$ are both selected, summand H becomes zero. This feature is used when data is merely transferred through the sum network with no arithmetic operations performed, as is the case in the interchange instructions and others.

Intermediate Functions. -- The intermediate functions comprise three high-speed gates per stage to produce the functions shown on Figure 2-13. These intermediate functions are used in the carry network and in the sum formation to be described. $R_n$ is also used as a source of the assertion form of $G_n$, when summand H is equal to $177777_8$, for the data path controlled by signal ESMTS+.

Carry Network. -- This network (see Figure 2-14 and LBD 117) is a succession of stages alternately forming the assertion and negation of the carry. The carry network implements the functions $C_n$ and $\overline{C}_{n-1}$,

where: $C_n = (G_n + H_n)(G_n H_n + C_{n+1})$, and

$$\overline{C}_{n-1} = (\overline{G}_{n-1} + \overline{H}_{n-1})(\overline{G}_{n-1}\,\overline{H}_{n-1} + \overline{C}_n)$$

Note that the ripple carry propagation and the new carry generation signals are not combined, but are made available on two and sometimes three wires. The carry signal is required in negation form from every stage and in assertion form from at least one of any

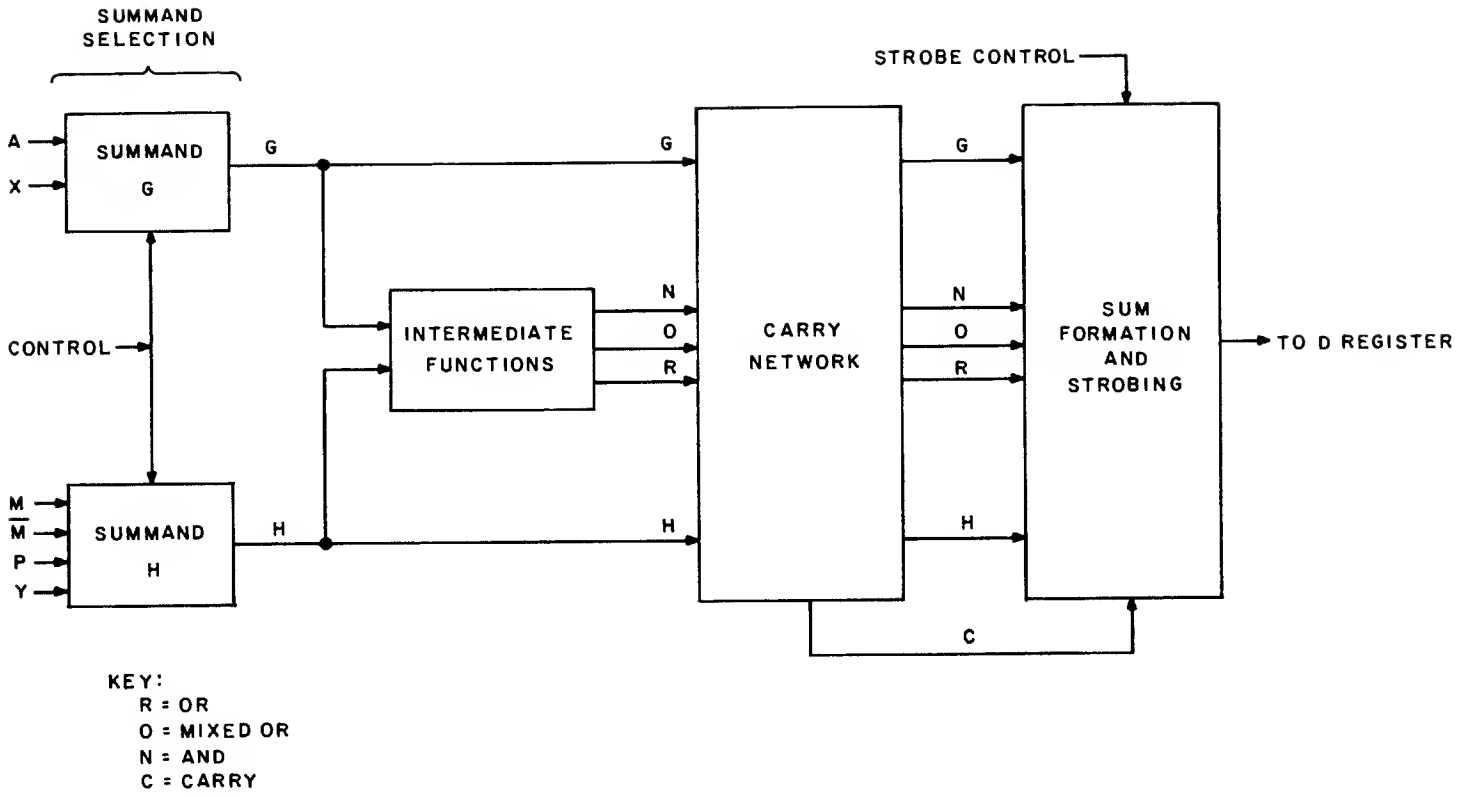



Figure 2-11. Sum Network Block Diagram

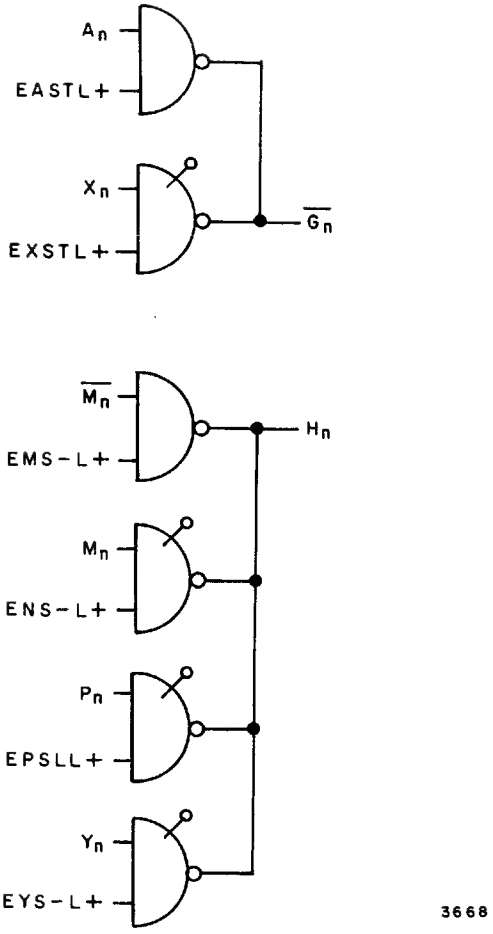



Figure 2-12. Summand Selection

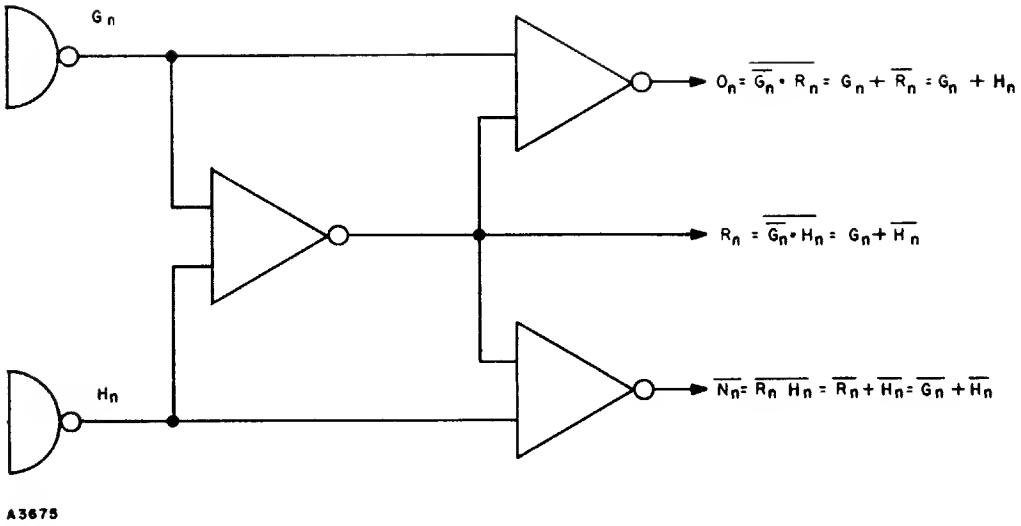



Figure 2-13. Intermediate Functions

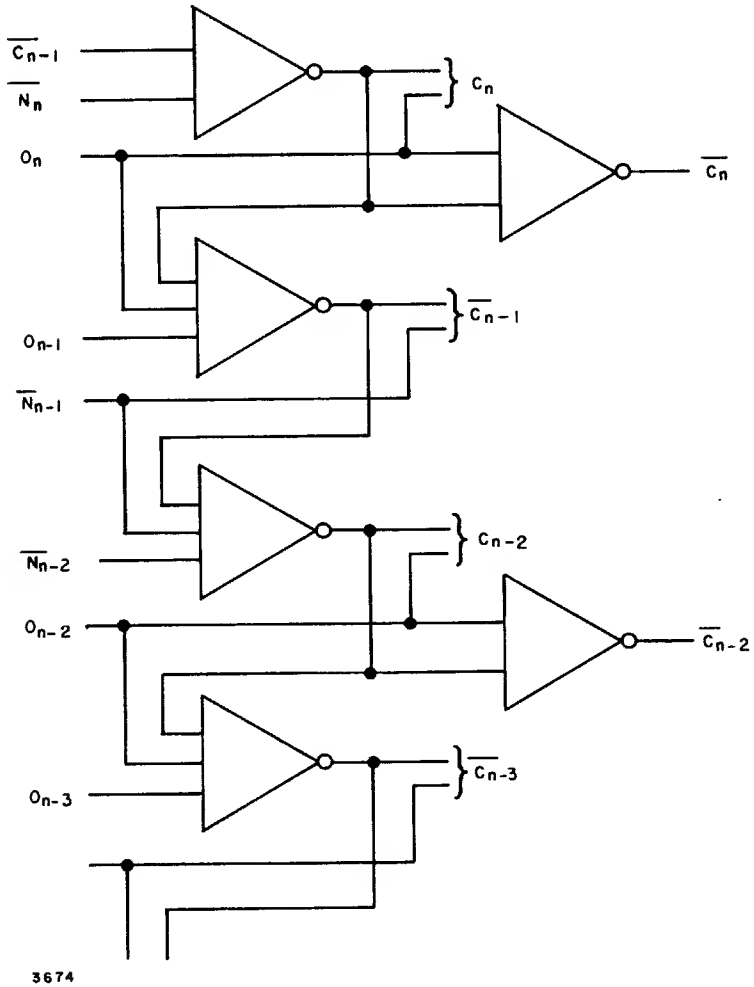



Figure 2-14. Carry Network Simplified Logic

two adjacent stages. The inverters at the right of Figure 2-14 complete this requirement without adding to the ripple delay.

To achieve even faster settling in the carry network it is necessary to anticipate the ripple carry at selected stages. This process is described with the following equations:

$$C_7 = (G_7 + H_7)\ (G_7 \cdot H_7 + C_8)$$

$$C_6 = (G_6 + H_6)\ (G_6 \cdot H_6 + C_7)$$

$$C_5 = (G_5 + H_5)\ (G_5 \cdot H_5 + C_6)$$

$$= (G_5 + H_5)\left[G_5 \cdot H_5 + G_6 \cdot H_6 + (G_6 + H_6)\, G_7 \cdot H_7 + (G_6 + H_6)\,(G_7 + H_7)\, C_8\right]$$

A similar anticipation is applied in the generation of the carry from stages 12, 8, and 3, as shown on Figure 2-15.

The least significant stage of the sum network (bit 16) provides for the injection of a 1 or a 0 as a pseudo-carry from a nonexistent 17th stage. This function (EIK17 from LBD 127) is used, for example, in the algorithm of the AOA (Add One to A) instruction. It is also widely used to offset the implicit - 1 value which summand H assumes when no selection of registers M, P, or Y is specified. The same function also completes the two's complementing action required in the TCA, SUB, and CAS instructions.

During some algorithms (CMA, TCA, ERA), the sum network is used for forming the bit-by-bit exclusive-OR of summand G and summand H, rather than their algebraic sums. For this purpose, signal JAMKN is applied to the intermediate function logic, the carry network, and the sum formation gates. The effect of a ground on this line is to suppress all carries (i.e., the output of each stage of the sum network is identical to that which would exist if the carry from the preceding stage was a logical 0).

Sum Formation and Strobing

The Boolean expression for the algebraic sum, S = G + H, can be manipulated into several equivalent forms:

$$S_n = \overline{G}_n \cdot \overline{H}_n \cdot C_{n+1} + \overline{G}_n \cdot H_n \cdot \overline{C}_{n+1} + G_n \cdot \overline{H}_n \cdot \overline{C}_{n+1} + G_n \cdot H_n \cdot C_{n+1}$$

$$\overline{S}_n = \overline{G}_n \cdot \overline{H}_n \cdot \overline{C}_{n+1} + \overline{G}_n \cdot H_n \cdot C_{n+1} + G_n \cdot \overline{H}_n \cdot C_{n+1} + G_n \cdot H_n \cdot \overline{C}_{n+1}$$

$$\overline{S}_n = \overline{G}_n\,(G_n + \overline{H}_n)\,\overline{C}_{n+1} + (G_n + H_n)\ (\overline{G}_n + \overline{H}_n)\ C_{n+1} + (G_n + \overline{H}_n)\, H_n \cdot \overline{C}_{n+1} \qquad (1)$$

Still another form of this expression is produced by noting, in the middle term of equation (1), that:

$$(\overline{G}_n + \overline{H}_n)\ C_n = (\overline{G}_n + \overline{H}_n)\ (G_n + H_n)\ (G_n \cdot H_n + C_{n+1}) =$$

$$(\overline{G}_n + \overline{H}_n)\ (G_n + H_n)\ C_{n+1}$$

Hence, $$\overline{S}_n = \overline{G}_n\,(G_n + \overline{H}_n)\,\overline{C}_{n+1} + (\overline{G}_n + \overline{H}_n)\ C_n + (G_n + \overline{H}_n)\, H_n \cdot \overline{C}_{n+1} \qquad (2)$$

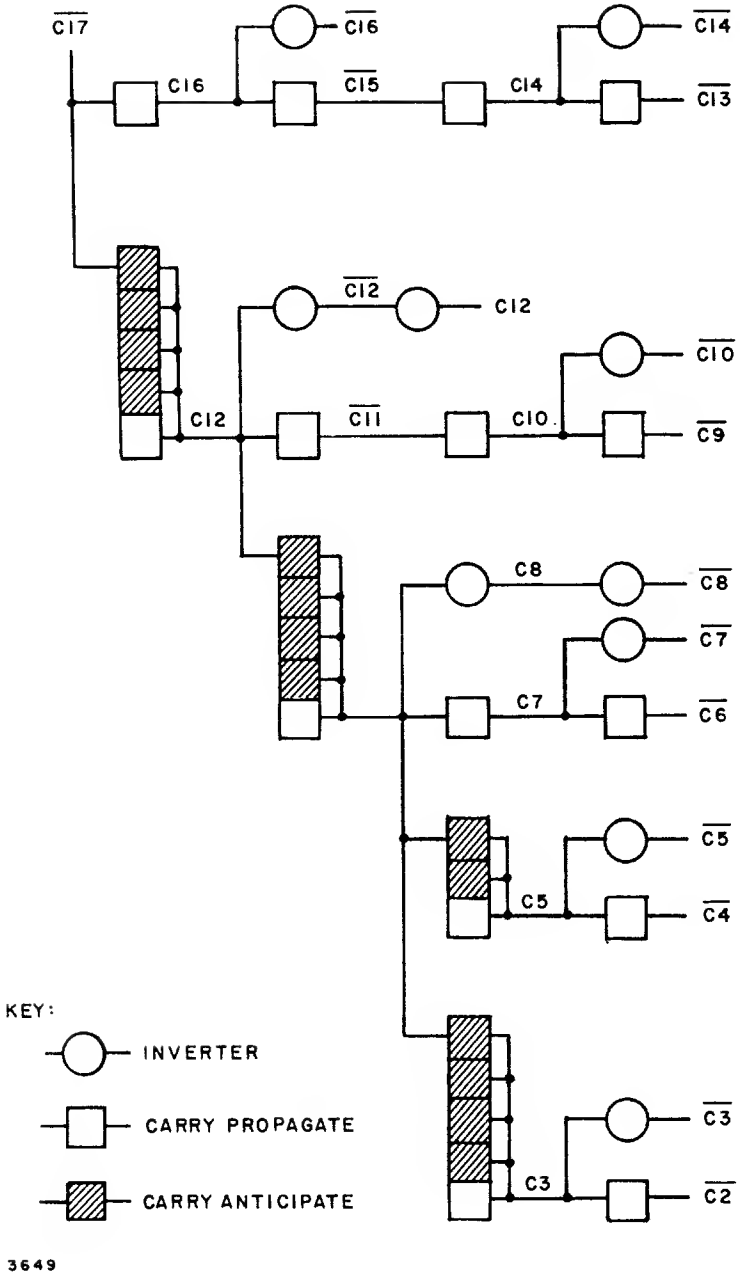



Figure 2-15. Carry Network Simplified Block Diagram

Equation (1) is used in the sum logic (Figure 2-16) of those stages (15, 13, 11, 9, 7, 6, 4, 2) for which the carry from the previous stage is available in true form; equation (2) is implemented in the other stages.

To satisfy timing requirements, the sum logic for stage 1 is extended by a process analogous to the carry anticipation discussed in the preceding section:

$$\overline{S}_1 = \overline{G}_1 (G_1 + \overline{H}_1) \overline{C}_2 + (G_1 + H_1) (\overline{G}_1 + \overline{H}_1) C_2 + (G_1 + \overline{H}_1) H_1 \cdot \overline{C}_2$$

where $C_2 = (G_2 + H_2) (G_2 \cdot H_2 + C_3)$

therefore,

$$\overline{S}_1 = \overline{G}_1 (G_1 + \overline{H}_1) \overline{C}_2 + (G_1 + \overline{H}_1) H_1 \cdot \overline{C}_2 + (G_1 + H_1) (\overline{G}_1 + \overline{H}_1) G_2 \cdot H_2 +$$

$$(G_1 + H_1) (\overline{G}_1 + \overline{H}_1) (G_2 + H_2) C_3$$

$$\overline{S}_1 = \overline{G}_1 (G_1 + \overline{H}_1) \overline{C}_2 + (G_1 + \overline{H}_1) H_1 \cdot \overline{C}_2 + (G_1 + H_1) (\overline{G}_1 + \overline{H}_1) (G_2 + \overline{H}_2)$$

$$H_2 + (G_1 + H_1) (\overline{G}_1 + \overline{H}_1) (G_2 + H_2) C_3 \qquad (3)$$

This is the function implemented on LBD 101.

Another special case appears on LBD 130, where the extended-sign-bit, $D_0$, is created by combining the carry, $C_1$, with extended summand signs, $G_1$ and $H_1$:

$$\overline{S}_0 = \overline{G}_1 \cdot \overline{H}_1 \cdot \overline{C}_1 + \overline{G}_1 \cdot H_1 \cdot C_1 + G_1 \cdot \overline{H}_1 \cdot C_1 + G_1 \cdot H_1 \cdot \overline{C}_1$$

But, $$C_1 = G_1 \cdot H_1 + (G_1 + H_1) C_2$$

$$\overline{C}_1 = \overline{G}_1 \cdot \overline{H}_1 + (\overline{G}_1 + \overline{H}_1) \overline{C}_2$$

when $$C_2 = (G_2 + H_2) (G_2 \cdot H_2 + C_3)$$

Hence, $$\overline{S}_0 = \overline{G}_1 \cdot \overline{H}_1 + \overline{G}_1 \cdot H_1 \cdot C_2 + G_1 \cdot \overline{H}_1 \cdot C_2$$

$$= \overline{G}_1 \cdot \overline{H}_1 + (\overline{G}_1 + \overline{H}_1) C_2$$

$$= \overline{G}_1 (G_1 + \overline{H}_1) + (\overline{G}_1 + \overline{H}_1) (G_2 + H_2) (G_2 \cdot H_2 + C_3)$$

$$\overline{S}_0 = \overline{G}_1 (G_1 + \overline{H}_1) + (\overline{G}_1 + \overline{H}_1) (G_2 + \overline{H}_2) H_2 + (\overline{G}_1 + \overline{H}_1) (G_2 + H_2) C_3 \qquad (4)$$

Addition. -- This paragraph contains a discussion dealing with the addition of two positive numbers, a positive and a negative number, and two negative numbers. These examples represent the three different combinations encountered in addition.

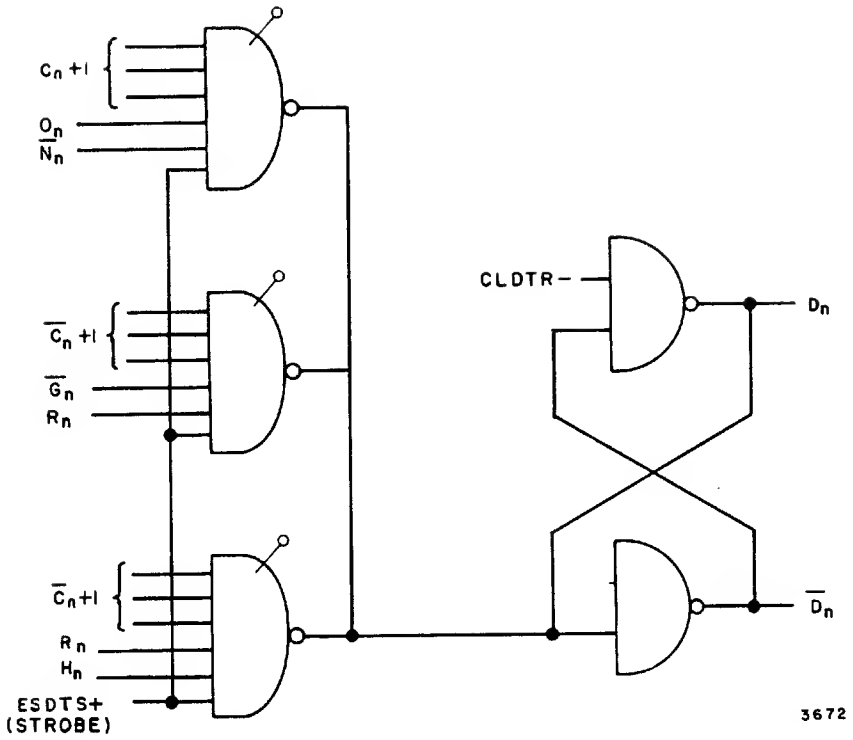



Figure 2-16. Sum Formation

Arithmetic operations in a two's complement oriented machine are logically easier to implement because the sign need not be considered, except to note overflow or underflow. The following examples show that, in two's complement arithmetic, only binary additions are required regardless of the sign of the data words.

For discussion purposes a 5-bit configuration is used (sign and four magnitude bits). The addition of two positive numbers is illustrated in Figure 2-17a. The contents of a memory location is stored in the M register and added to the contents of the A register. The addition occurs in the sum network. The sum of the two numbers is transferred to the A register via the D register.

The next case, a positive and a negative number, is equally simple (Figure 2-17b). For this example the numbers +7 and -12 are to be added, the latter is in the A register at the start of the addition.

All that needs to be done is to add A to the effective operand in memory (+7). The operand is transferred to the M register and presented in summand H. The contents of A is presented to summand G. The sum is in two's complement (-5).

Adding two negative numbers is no more difficult since both numbers are in two's complement. (Refer to Figure 2-17c.) The adding consists of presenting the contents of M to summand H (two's complement of -5) and presenting the contents of A to summand G (-9). The resultant sum is in two's complement (-14).

| | | | | | |
|---|---|---|---|---|---|
| 00111 | (+7) | 00111 | (+7) | 10111 | (-9) |
| 00101 | (+5) | 10100 | (-12) | 11011 | (-5) |
| 01100 | (+12) | 11011 | (-5) | 10010 | (-14) |
| (a.) | | (b.) | | (c.) | |

Figure 2-17. Addition Examples

Subtraction. -- Two's complement subtraction is quite simple (see Figure 2-18). One of the numbers (the contents of M) is two's complemented prior to being added to the contents of A. This occurs at the input of summand H with selection signals ENSHL and ENSLL, and carry injections equal EIK17. The sum is transferred to D to provide the result directly to A.

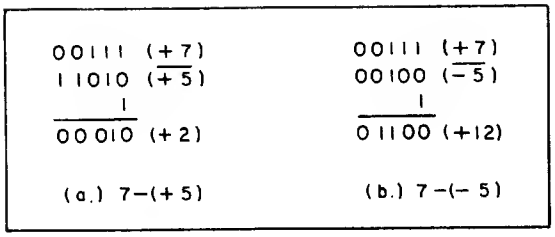



Figure 2-18. Subtraction Examples

Master Clock

All instructions are performed under the control of a two dimensional time grid. The two dimensions are fine and coarse. The fine dimension is controlled by the timing level generator (TLG). The coarse dimension is controlled by the phase register (discussed later).

The master clock oscillator (MCO) is the heart of the time grid. The MCO (LBD 118) generates three output waveforms: MCSET, MCRST, and MCTLG. The MCO is controlled by the start-stop logic (LBD 126) via the run flip-flop (RUNFF). When the computer is initialized, RUNFF is cleared and the MCO is at rest. Setting RUNFF starts the MCO through a five-epoch cycle. (See Table 2-1 and Figure 2-19.)

Table 2-1.
MCO Periods

| Nominal Duration (ns) | MCSET | MCRST | MCTLG |
|---|---|---|---|
| 170 | 0 | 0 | 0 |
| 40 | 1 | 1 | 0 |
| 100 | 1 | 1 | 1 |
| 40 | 1 | 0 | 1 |
| 50 | 1 | 0 | 0 |

The MCO continues to cycle until RUNFF is cleared with the MSTR CLEAR button on the control panel or with a programmed halt instruction (HLT). If the RUNFF is cleare during the MCRST pulse, no discontinuity is introduced into the MCO output waveforms. The MCO completes the cycle in progress and stops at the end of the fifth epoch (the trailir edge of MCSET).

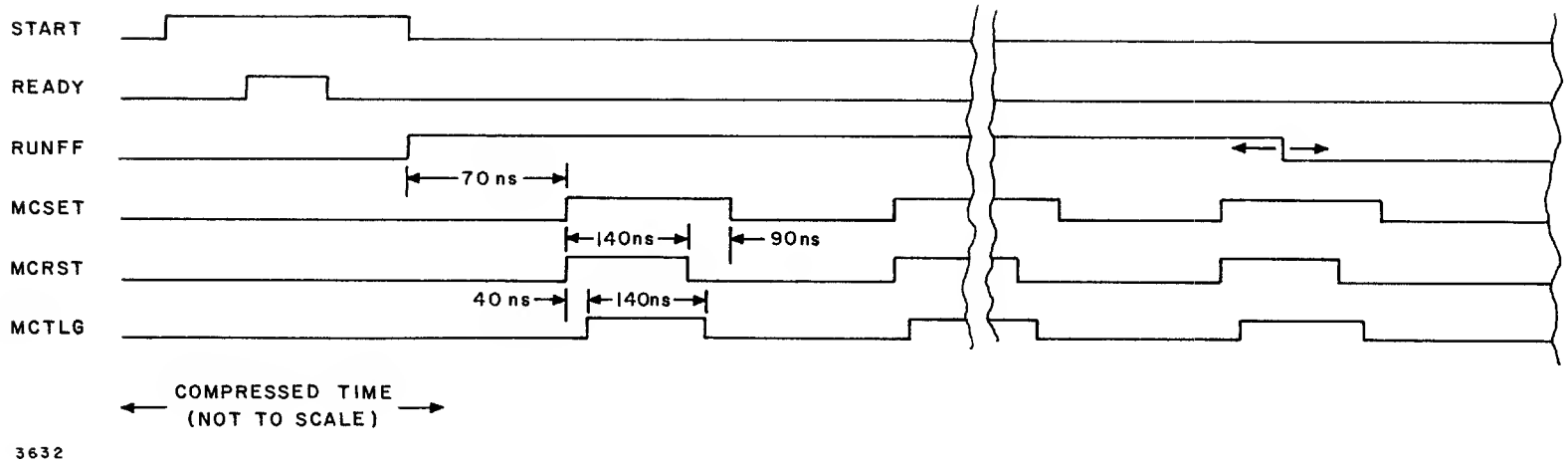



Figure 2-19. Master Clock Waveforms

Timing Levels. -- The TLG controls the fine dimension of the timing grid (LBD 118). Normally, the system cycles sequentially through four timing levels: TL1FF, TL2FF, TL3FF, and TL4FF. Only one of these four levels is present at any time and the level changes at the end of each MCO cycle. When the system is initialized (see Figure 2-9), the TLG is preset such that only timing level 3 (TL3) is present. Auxiliary flip-flops TL13F, TL23F, and TL24F are set. When the MCO is started, each MCTLG pulse changes the auxiliary flip-flops in accordance with the current primary timing level. Thus, TL13F is set during TL1 and reset during TL3, TL23F is set during TL2 and reset during TL3, and TL24F is set during TL2 and reset during TL4. The states of the auxiliary flip-flops are then used (at the trailing edge of MCSET) to control the transition to the next timing level. (See Figure 2-20.)

During the execution of certain instructions (shifts, MPY, DIV, NRM, TCA, and HLT) the sequence of timing levels is modified. At these times the transition from TL3 to TL4 is blocked and the TLG returns to TL2. This is a function of signal RPTT2 (repeat TL2) (Figures 2-20 and 2-21).

## Phase Register

The phase register (LBD 119 and Figure 2-22) controls the second of two dimensions (coarse) of the timing grid described under the master clock. Three phases are sufficient for all central processor instruction sequences; the cycles are Fetch (F), Indirect (I), and Execute (A). Each cycle starts with TL1 and ends with TL4. The duration of a phase is thus at least four clock cycles. It can be longer for the following reasons.

If signal RPTT2 is present, TL2 and TL3 are repeated for a total of at least six clock cycles (TL1, TL2, TL3, TL2, TL3, TL4). Certain A-cycle instructions are terminated with the second TL4 rather than the first TL4. This is a function of the contents of the shift counter. During the first pass through TL4, the shift counter does not equal zero; the A cycle is terminated on the second pass when the shift counter is forced to zero. (See shift counter discussion.) The last example occurs during an indirect cycle when it is uninterrupted during multi-level indirect addressing.

Two versions of each phase are generated, an early and late cycle. For example, the F cycle consists of FCYEF and FCYLF (F cycle early and late, respectively). FCYEF is established during TL4 of the previous cycle and is available for use in controlling actions during TL1 and TL2. FCYLF is copied from FCYEF during TL1 for use in controlling actions during TL3 and especially TL4. Some exceptions are made to these rules during TL2 and TL3 to equalize loading on the phase flip-flops.

## Shift Counter

The shift counter (LBD 121) is a 6-bit counter that operates in conjunction with the phase register to extend the execution of those instructions requiring more time. The F cycle is extended for shift, TCA, and HLT instructions. The A cycle is extended for JST, CAS, IRS, LDX, and others.

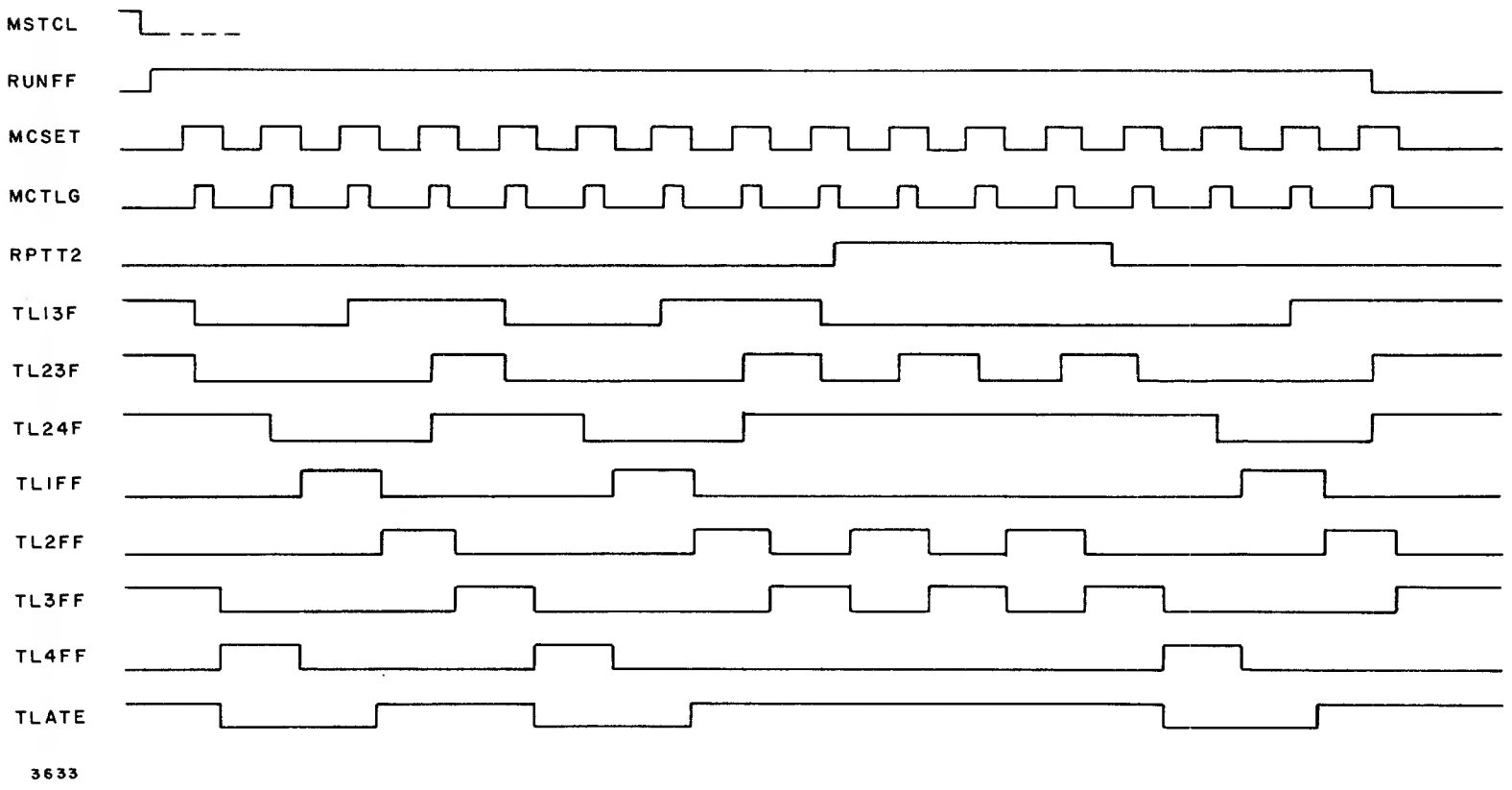
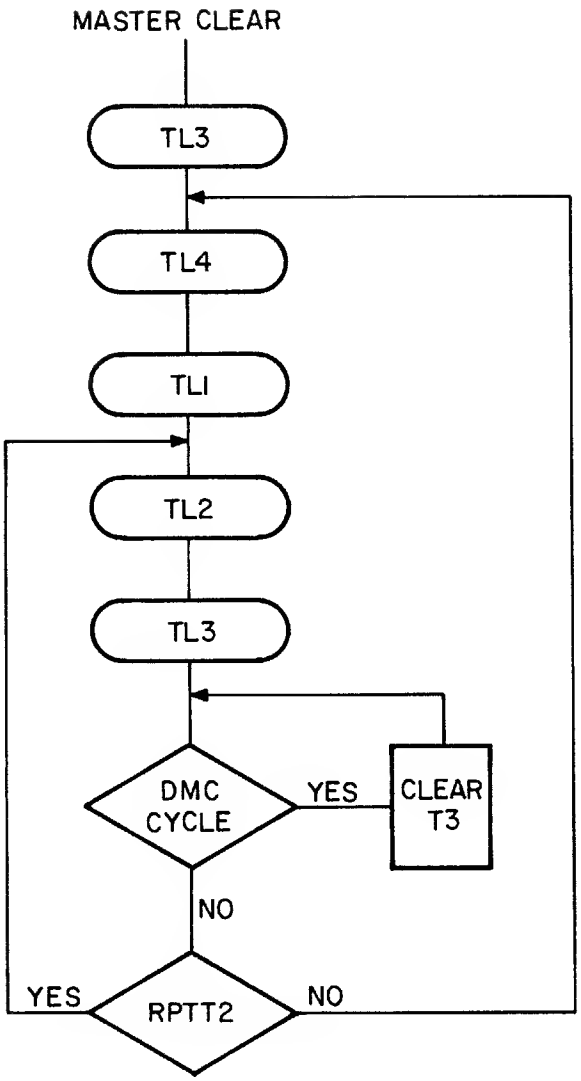



Figure 2-20. Master Clock Waveforms




5840

Figure 2-21. Timing Level Generator Flow Chart

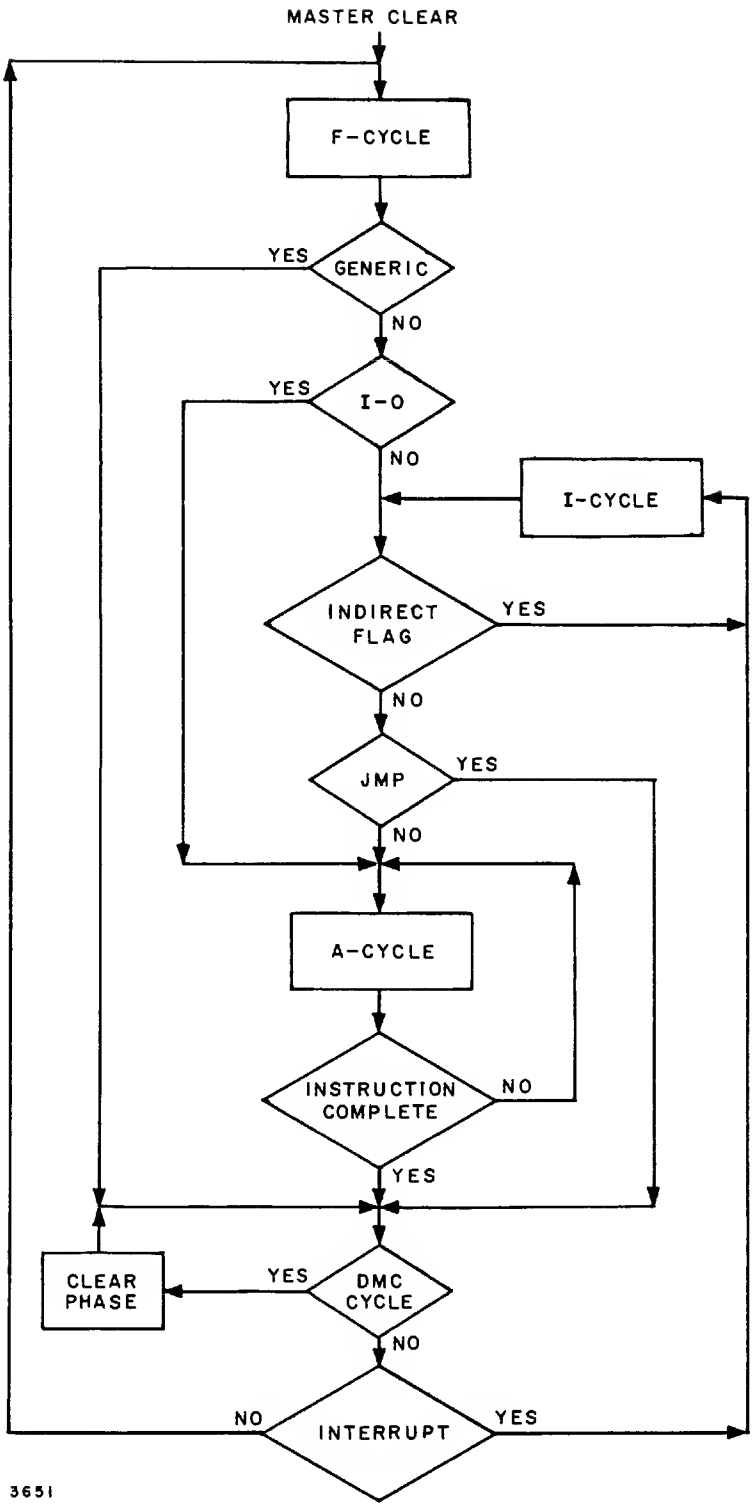



Figure 2-22. Phase Register Flow Chart

During TL1 of the fetch cycle of each central processor instruction, the shift counter is cleared to zero and remains in this state throughout the majority of operations. However, during the first TL3 of a shift instruction, for example, the shift counter is loaded from the instruction address field. This reflects the two's complement of the number of places to be shifted. At the end of TL3, the non-zero content of the shift counter enables the generation of control signal RPTT2 (repeat TL2) as previously described. TL2 is repeated as many times as is required to complete the designated number of shifts. Thus, the shift counter is responsible for determining the duration of the instruction (1.6 $\mu$s plus 0.8 $\mu$s per shift).

## Data Storage

The various data storage registers in the central processor are formed with cross-coupled gates. With rare exceptions, all data transfers into these registers are performed by clearing all bits of the registers and then setting selected bits of the register to the desired state. (The D, E, and X registers are cleared by setting them to all 1's.)

These two steps are actually carried out in overlapping fashion using the MCRST and MCSET master clock signals. Figure 2-23 is a simplified logic diagram of a typical data transfer depicting the control and timing of these paths. The numbers in parenthesis denote the latest times at which various key signals stabilize (measured in nanoseconds from the end of the previous MCO cycle). The clearing (MCRST) and setting (MCSET) signals reach the receiving register simultaneously. Proper operation is ensured by the earlier termination of MCRST, combined with the common collector connection of the set gate to the flip-flop. (See Figure 2-23.)

## Operation Decoding

The output of the F register is used at the input of two binary-to-octal decoders (LBD 120) to develop signals for the various op codes used in the central processor. The F register is loaded from the M register during an F cycle. With reference to Table 2-2, note that bit F03 enables only one of the two decoders at a time. Bits F04, F05, and F06 determine the specific op code.

Certain similar instructions are grouped for convenience in the H316. These groups are:

a. OPG00
b. OPG3C
c. OPGAA
d. OPGDP
e. OPGJS
f. OPGMD
g. OPGNS
h. OPGSM
i. OPGWR

OPG00 instructions have 0's in bit positions M03 through M06. OPG3C instructions are three-cycle memory reference instructions. These include JST, IRS, CAS, IMA, LDX, and double-precision instructions. The OPGAA instructions are those memory reference instructions that utilize the A register during an A cycle (with some exceptions). These

include LDA, ANA, ERA, ADD, SUB, and IMA. Instructions utilizing double-precision arithmetic operation fall into the OPGDP group. These are ADD, SUB, LDA, and STA. OPGJS are those that jump or skip such as the JMP, JST, IRS, and skip enabled instructions.

Instructions involved with negative sums belong to the OPGNS instructions. They are SUB, IRS, and CAS. Instruction CAS satisfies these conditions only when $(A)_1$ equals $(M)_1$.

STA, IMA, LDX, and STX belong to group OPGSM. This group deals with a sum to M register control as its common point. A write read control group, OPGWR, includes instructions STA, IMA, LDX, STX, IRS, and JST.

Table 2-2.
Op Code Decoding

| Op Code | F03 | F04 | F05 | F06 | |
|---|---|---|---|---|---|
| IMAOP- | 1 | 0 | 1 | 1 | (LBD 120) Location J2-J9 |
| IRSOP- | 1 | 0 | 1 | 0 | |
| CASOP- | 1 | 0 | 0 | 1 | |
| JSTOP- | 1 | 0 | 0 | 0 | |
| DIVOP- | 1 | 1 | 1 | 1 | |
| MPYOP- | 1 | 1 | 1 | 0 | |
| LSXOP- | 1 | 1 | 0 | 1 | |
| IOGRP- | 1 | 1 | 0 | 0 | |
| ANAOP- | 0 | 0 | 1 | 1 | (LBD 120) Location F2-F9 |
| LDAOP- | 0 | 0 | 1 | 0 | |
| JMPOP- | 0 | 0 | 0 | 1 | |
| OPG00- | 0 | 0 | 0 | 0 | |
| SUBOP- | 0 | 1 | 1 | 1 | |
| ADDOP- | 0 | 1 | 1 | 0 | |
| ERAOP- | 0 | 1 | 0 | 1 | |
| STAOP- | 0 | 1 | 0 | 0 | |

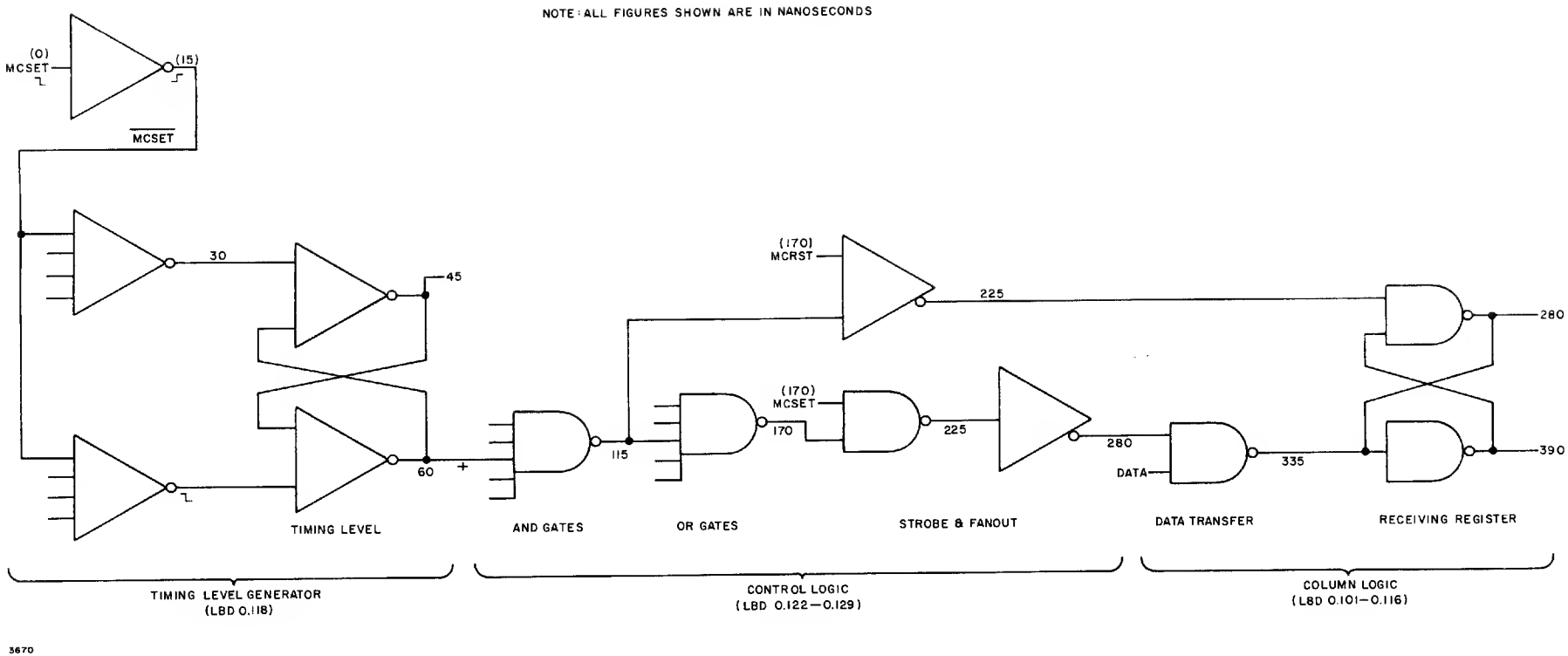



Figure 2-23. Clock Timing of Typical Data Transfer

## INPUT/OUTPUT TELETYPE INTERFACE

This paragraph contains a discussion of the ASR-33/35 teletype interface logic. The interface logic is designed to function with either the ASR-33 or ASR-35. Either model of the ASR and the interface logic provide the capability for reading paper tape, punching paper tape, generating hard copy from computer data, and providing a keyboard input to the computer. The teletype can be used in either on-line or off-line modes. The abbreviation ASR as used herein refers to either the ASR-33 or ASR-35.

The ASR is an I/O device that transmits data serially over a two-wire line. One line is the signal line and the other is the return line. When transferring a character from the ASR to the computer, the character is shifted serially over the signal line into a 9-bit buffer register. The transfer from the buffer register to the computer is in parallel (input mode). When a character is transferred from the computer to the ASR, the process is reversed. The character is transferred in parallel via the output bus to the buffer register. The character is shifted through the buffer register and out over the signal line to the ASR (output mode).

An ASR character consists of an 11-bit code made up of marks and spaces analogous to logic 1 and 0. Approximately 65 ma in the signal line constitutes a marking condition. No current flow indicates a space. The quiescent state of the ASR is the marking condition.

The first bit of an 11-bit character code is the start pulse and is always a space. Bits 2 through 9 are the information bits and can be any combination of marks and spaces. Bits 10 and 11 are always marks and denote the end of a character transmission.

NOTE

All referenced LBDs are contained in the H316 Central Processor Instructions and Diagrams Manual.

Figure 2-24 is a simplified block diagram of the interface logic as contained on LBDs 340, 341, and 342. The two-phase clock is started and stopped as a function of a busy flip-flop in the sequence control. The mode control logic sets the interface into an input or output mode as determined by the state of ADB10. The address decoder generates a teletype address signal whenever the address field of an I/O instruction is XX04.

The sequence control is a group of flip-flops and associated gates which ensure the proper sequence of events within the interface. Note that the status of the buffer register influences operation within the sequence control, as well as the generation of shift pulses. The exact function and the buffer status are described in later text.

The buffer register is a combination serial and parallel shift register. When transferring data in from the signal line, character information is transmitted via the ASR encoder/decoder. Transferring data out to the ASR is routed through the buffer register and ASR encoder/decoder.

The SKS logic generates the device ready signal in response to any of the ASR SKS instructions. (See Programmers Reference Manual, Doc. No. 70130072156A.) The SMK logic generates a program interrupt request when the interface is ready and the SMK flip-flop is set.

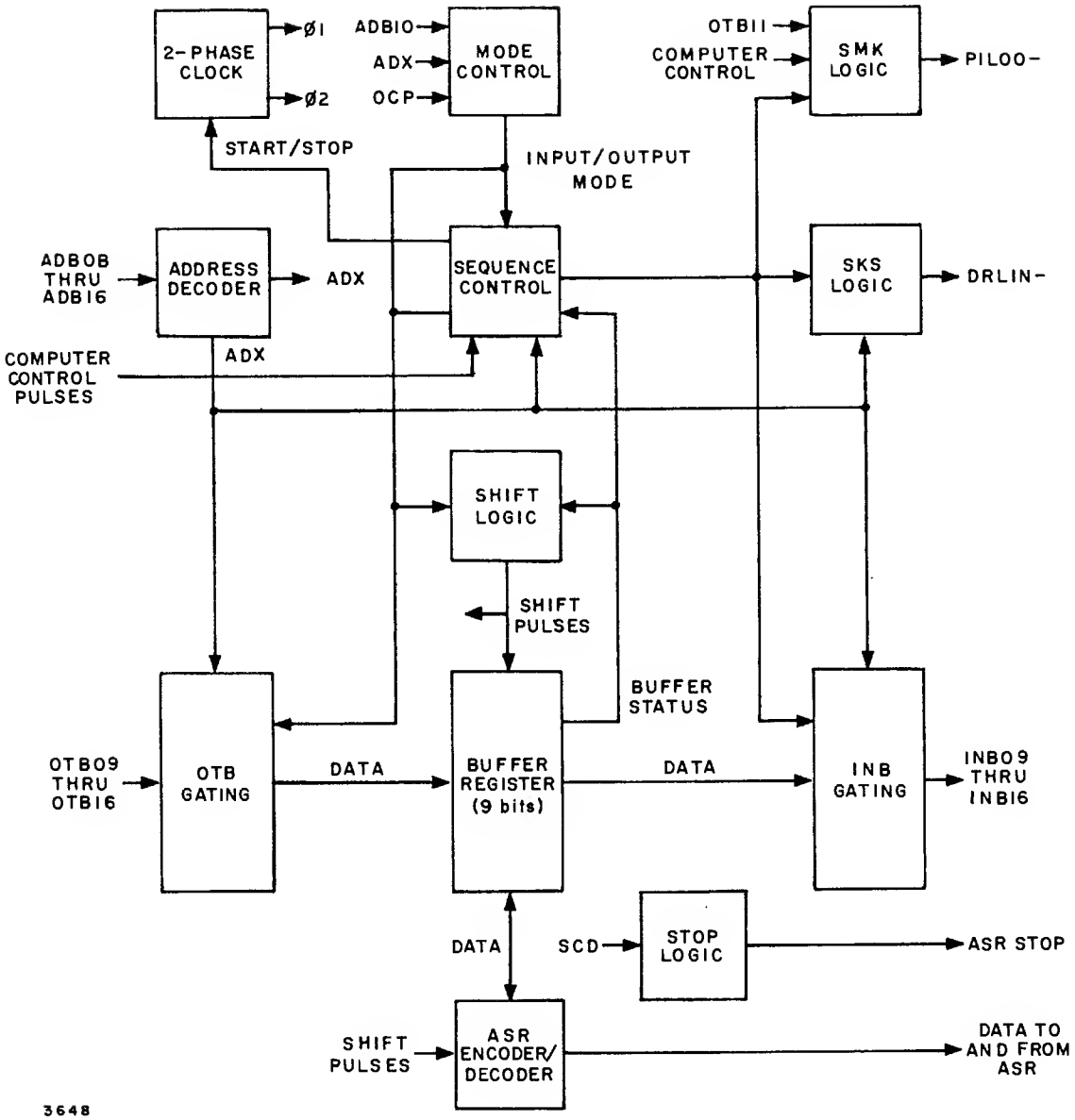



Figure 2-24. ASR Interface Block Diagram

The stop logic monitors the contents of the buffer register for an X-OFF character. When this character is present, a stop flip-flop is available for one character time for program test.

## OPERATION

The following detailed discussion contains a description of interface operation in the input and output modes, each based on a timing diagram and LBDs 340, 341, and 342. The input mode discussion is given first.

### Input Mode

Assume that an OCP '0004 (Enable ASR in Input Mode) is issued. As a result of this instruction, the interface logic receives an OCPLS- signal and a teletype address code (see LBD 342). The OCPLS- signal is used to generate signal OCPXX+. The address code generates signal TYADX-.

OCPLS- is used, in conjunction with TYADX+ and ADB10-, to generate signal TYICP- and to reset the output mode flip-flop TYOUT, LBD 342 E5. TYICP- generates PREST- which clears the buffer register and resets the TYRDY, TYK0X, TYK1X, and TYCFB flip-flops. This initializes the interface for input mode operation.

The next step is for the operator to strike a key on the ASR. This action causes TYDAT- to become passive (LBD 341 C9), and in turn causes TYDTA+ to become active. (See Figure 2-25 for input mode timing diagram.) This sets the busy flip-flop (TYBSY, LBD 340 P2). With TYBSY set, the clock is started (LBD 340 A1).

Note that all the conditions for the generation of the first of the two-phase clock pulses, TYK1P, are present at the input of gate TYCLK (LBD 340 C2). The trailing edge of TYK1P+ sets the TYCFA flip-flop in conjunction with signals TYSTP- and TYCAL+.

With reference to Figure 2-25, note that as the clock cycles the TYK0X flip-flop is reset, enabling the generation of the second of the two-phase clock pulses, TYK2P (LBD 340 C12). With all inputs to gate TYCLK true, the first of nine shift pulses is generated (TYSFT). The function of this pulse is two-fold. First, it shifts data bits into the buffer register, and second it keeps track of the number of data bits shifted into the buffer register.

The first shift pulse always stores a space in the LSB of the buffer register. (When this space is propagated through the buffer register it is stored in the TYDR0 flip-flop, a condition necessary to complete the loading and transfer of one teletype character.) After the generation of the first shift pulse, notice that the timing follows the pattern just described. The only thing that differs is that each new data bit generated (a total of eight for each character) is entered into TYDR0 and the previously entered data bits are propagated down the buffer register, with the lower order stages copying the next highest order stage.

After the ninth shift pulse, the space has been propagated through the buffer register and stored in TYDR0. The change in the state of TYDR0 resets the TYCFA flip-flop, in conjunction with TYOUT- and TYK1P+, inhibiting the generation of additional shift pulses. Further, with both TYCFA and TYCFB reset, the TYSTP flip-flop is set on the trailing

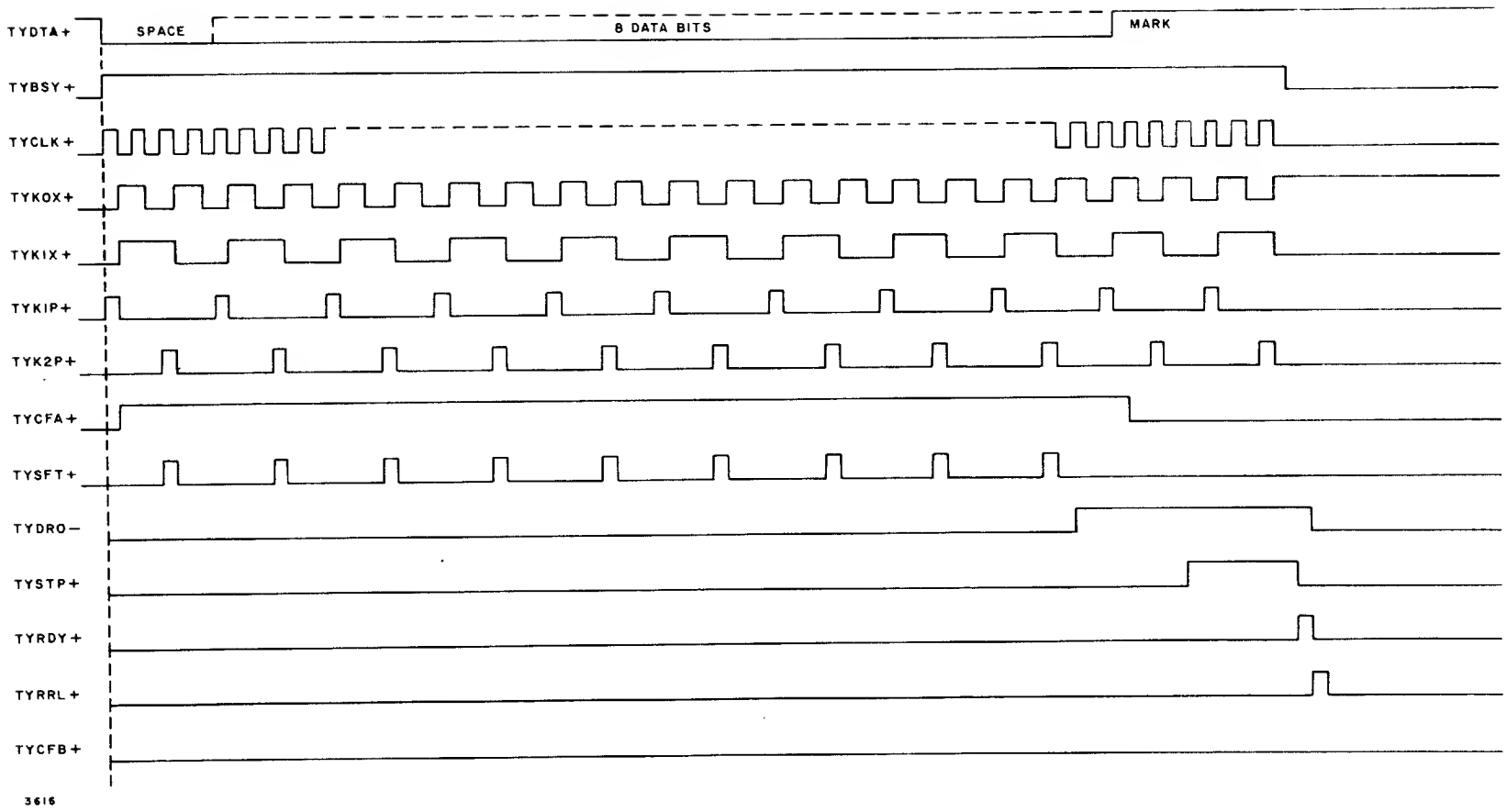



Figure 2-25. Input Mode Timing Diagram

edge of TYK2P+ (LBD 340 K7). Concurrent with the generation of the ninth shift pulse, TYDTA is forced to a mark condition. Since the TYSTP flip-flop is set, conditions are present at the input of the busy flip-flop to reset it at the trailing edge of TYSTP+, which stops the clock and sets the ready flip-flop TYRDY.

The CPU detects that the ASR has information to transfer in one of two ways: either by SKS '0004 or SKS '0204 (Skip if ASR is Ready in ASCII Mode, or Skip if ASR is Ready in Binary Mode) or by program interrupt on the PIL00- line if mask flip-flop TYMSK is set (LBD 342). As a result, either of four INA instructions is given (INA '0004, '1004, '0204, or '1204) and the data is strobed into the CPU. The CPU signals the ASR that it will accept the data by generating signal RRLIN-.

RRLIN+ and TYADX+ generate signal TYRRL- which resets the ready flip-flop (LBD 342). TYADX- is used to strobe the contents of the buffer register onto the input bus (LBD 341). The interface is now ready for the next character from the ASR.

Output Mode

The output mode discussion is entered with the generation of an OTA. This is done to avoid complicating the discussion with events that occur prior to the generation of the OTA (during cycle).

Assume that an OTA '0004 is issued. A function of this instruction is to generate signal RRLIN-. RRLIN-, in conjunction with TYADX+, generates TYRRl- and, in conjunction with TYADX+ and TYOUT+, generates TYOTP-. TYRRL- resets the ready flip-flop (TYRDY, LBD 342). (See Figure 2-26 for timing diagram.)

TYOTP- performs several functions. First, it resets TYDR0 in the buffer register (LBD 341). When TYDR0 is reset, the conditions for generating TYRCF- are no longer present to hold TYCFA reset. Then, it generates TYTCP- which sets the busy flip-flop (TYBSY, LBD 340). TYOTP+ is used to load the data from the output bus into the buffer register (LBD 342). The trailing edge of TYOTP+ sets the TYCBC flip-flop, making TYDTA+ true.

With both TYDR0 and TYCFB reset, conditions are present to set TYCFA with the first TYK1P+ (see TYCAL, LBD 340). Further, note that the inputs to signal driver/receiver TYSIG- (LBD 341-C9) are both 0, causing a space to be sent to ASR (TYSIG- drops to less than 3 mA, the space condition).

A great deal has occurred in the previous paragraphs. First, the ready flip-flop (TYRDY) was reset to inform the CPU that the interface is involved in a data transfer. This condition is tested by an "if ready" SKS. Second, the busy flip-flop (TYBSY) is set. The condition is tested by an "if not busy" SKS. Now the CPU knows that the ASR is both "busy" and "not ready" and cannot accept or provide data until the present operation is terminated.

The busy flip-flop enabled the clock to run, permitting the sequence control logic to initialize prior to the acceptance of data from the output bus. When the character is transferred from the CPU to the interface, the OTB gating logic stores the character in the buffer register, enabled with internal control signal TYOTP+.

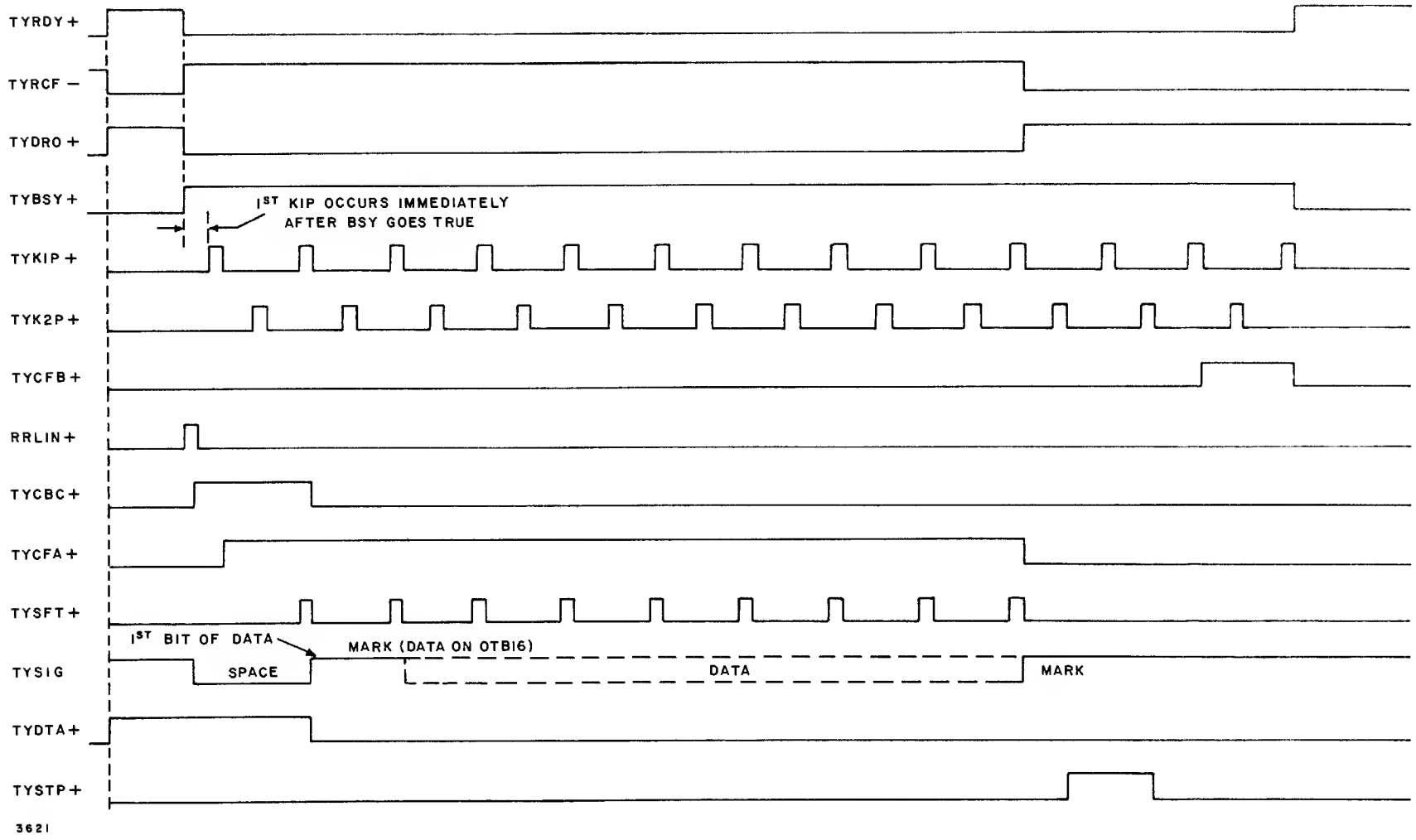



Figure 2-26. Output Mode Timing Diagram

Certain other events must now occur. The buffer register contains a character that needs to be transferred to the ASR. Since the ASR input must be serial, some means of serial entry must be provided. Further, it is important to keep track of the data bits in order to determine when the last bit has been routed to the ASR. This is implemented by shifting the data bits, one at a time, into the TYDR0 flip-flop. Since the signal driver/receiver is sensitive to the state of TYDR0, each bit shifted into TYDR0 is sent to the ASR as a mark or space. As the data bits are shifted through the buffer register, 0's are pushed into the buffer register, one at a time, as a function of the shift pulses and the reset state of the TYCBC flip-flop. Now let time start again to generate the first shift pulse.

In examining the inputs to gate TSFTB- (LBD 340-67), note that the states of these inputs are such that signal TYSFT+ (shift pulse) is generated. TYSFT+ shifts the contents of the buffer register down one position and, on its trailing edge, resets the TYCPC flip-flop. (See LBD 341.) This unconditionally puts a 1 in. TYDR8 and sends the first data bit to the ASR via the signal driver/receiver TYSIG- (LBD 341-C9). Each successive shift pulse enters a 0 into TYDR8 and shifts the data bits through the buffer register to the ASR.

With the generation of the ninth shift pulse, note that the inputs to gates TYCRF-, TY36R+ reflect the contents of the buffer register. This causes signal TYRCF- to be generated which in turn resets the TYCFA flip-flop, inhibiting the generation of additional shift pulses. As a result, a marking condition is presented to the ASR.

The operation is terminated when the TYSTP and TYCFB flip-flops are set. The mutual dependence of these two flip-flops causes them both to become reset after the combination of TYCFB and TYDR0 resets the busy flip-flop which in turn stops the clock.

## Dummy Cycle

The dummy cycle is necessary to give the ASR enough time to respond to a change from input to output mode. Figure 2-27 is a timing diagram that illustrates what happens in the interface when an OTA is given immediately after an OCP and when an OTA is given after the dummy cycle. Observe that when the OCP is issued, the busy flip-flop is set and the clock is started. The OCP also causes TYDR0 to be set and the remainder of the buffer register is reset. This makes TYRCF- true, resetting the TYCFA flip-flop (LBD 340).

If the OTA is given immediately following the OCP, conditions are as described in the output mode discussion. However, if the OTA is given after the interface has generated a second TYK1P clock pulse, the busy flip-flop is reset, stopping the clock. (See dashed lines on Figure 2-27.) In this second case, the clock is restarted by the OTA and the interface is re-initialized and ready to process the character from the CPU.

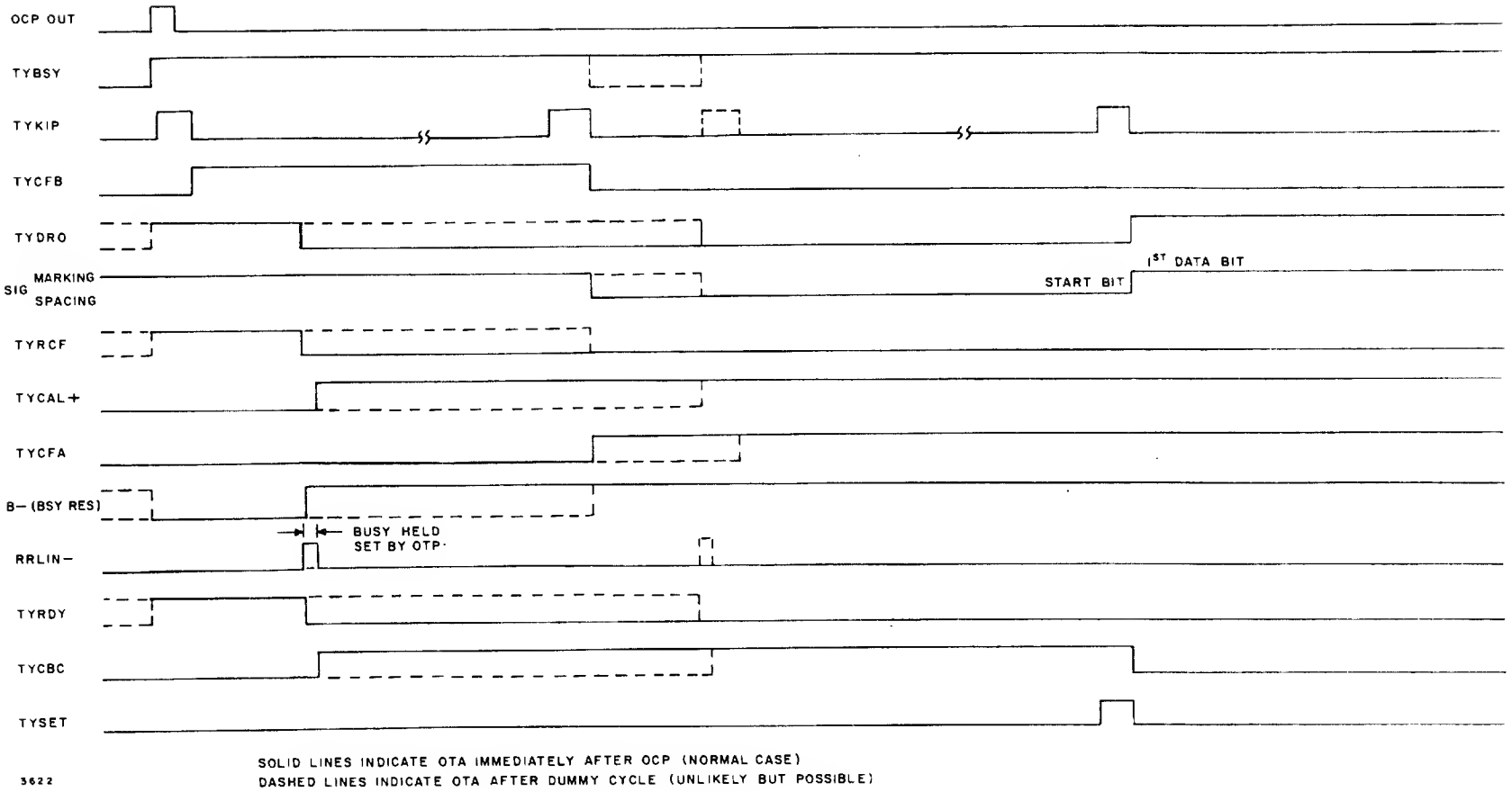



Figure 2-27. Dummy Cycle Timing Diagram

Cut Along Line

## FIELD ENGINEERING MANUAL
## USERS' REMARKS FORM

TITLE:

DOC. PART NO. _____

DATED _____

ERRORS NOTED:

Fold

SUGGESTIONS FOR IMPROVEMENT:

Fold

DATE _____

FROM: NAME _____

COMPANY _____ M/S _____

TITLE _____

ADDRESS _____

_____ ZIP _____

FIRST CLASS
PERMIT NO. 39531
WELLESLEY HILLS
MA. 02181

**BUSINESS REPLY MAIL**

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY

HONEYWELL INFORMATION SYSTEMS
FIELD ENGINEERING DIVISION
141 NEEDHAM STREET
NEWTON HIGHLANDS, MA. 02161

ATT'N: FIELD ENGINEERING PUBLICATIONS

**Honeywell**

The Other Computer Company:

**Honeywell**

HONEYWELL INFORMATION SYSTEMS